ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

ADAPTACIÓN DE UNA APLICACIÓN LIBRE DE GESTIÓN
PARA LA EMPRESA

Guillermo Gutiérrez Herrera

15 de febrero de 2012

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

ADAPTACIÓN DE UNA APLICACIÓN LIBRE DE GESTIÓN PARA LA EMPRESA

- Departamento: Lenguajes y Sistemas Informáticos
- Directores de proyecto: Manuel Palomo Duarte & Juan Boubeta Puig
- Autor del proyecto: Guillermo Gutiérrez Herrera

Cádiz, 15 de febrero de 2012

Fdo: Guillermo Gutiérrez Herrera

Agradecimientos

Me gustaría agradecer el apoyo que me ofrecieron mis padres, hermanos y abuelos, y a ellos les dedico este Proyecto Fin de Carrera.

Gracias.

Licencia

Este documento ha sido liberado bajo Licencia GFDL 1.3 (GNU Free Documentation License). Se incluyen los términos de la licencia en inglés al final del mismo.

Copyright (c) 2012 Guillermo Gutiérrez Herrera.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Índice general

Índice de figuras

CAPÍTULO 1

Introducción

Este documento describe el proceso que hemos seguido para la ejecución del proyecto. Comenzamos con la toma de requisitos, cuyo resultado final es la Especificación de Requisitos del Software (SRS), realizada a partir de las conclusiones obtenidas tras las conversaciones con el cliente, y de analizar el sistema actual. A continuación pasamos a diseñar los componentes software que debemos añadir al sistema para que cumpla con los requisitos identificados de la etapa anterior. En los siguientes capítulos mostramos los detalles de la solución implementada y las pruebas realizadas al sistema para validar que el resultado final satisface las necesidades del cliente.

El cliente es TITANIA*, Ensayos y Proyectos Industriales* [28], una empresa mediana, cuya principal actividad es realizar ensayos a materiales industriales, en su mayoría relacionados con el sector aeronáutico. Sintetizando, podemos decir que en cada trabajo, reciben una o varias muestras del material, que se estudia, analiza y somete a un conjunto de ensayos solicitados por su cliente. El técnico que completa el ensayo rellena un informe detallando las conclusiones obtenidas, y si el material se puede considerar apto o no según una determinada normativa. Una vez completada la prueba, se puede devolver el material junto con la entrega del informe, siempre que la prueba no sea destructiva.

1.1. Situación previa

El primer paso para llevar satisfactoriamente a cabo una informatización de una empresa es conocer su funcionamiento actual. En el caso de Titania, se dispone de una red de equipos Windows, centralizando una base de datos MS Access en un servidor local. La estructura de esta base de datos ha ido creciendo y haciéndose más compleja cada vez, según han ido necesitándose cambios. El problema principal de usar Access es la compartición de la información con el modo exclusivo de escritura. Para realizar inserciones o actualizaciones en la base de datos, se requiere de la comunicación oral entre los empleados para solicitar acceso exclusivo y así garantizar la integridad de los datos almacenados.

Esto implica una serie de limitaciones al personal que va en detrimento de la productividad. Por ejemplo, dos técnicos diferentes trabajando simultáneamente en distintas partes del mismo informe deberán editarlo de manera secuencial, cuando no debería de existir ningún impedimento técnico para hacerlo en paralelo.

Además, la estructura de la base de datos no está suficientemente normalizada como para evitar duplicidad e incoherencias que se producen típicamente en estos sistemas.

Resumiendo, los problemas pendientes de resolver son:

- El desorden estructural y la desnormalización existente en la base de datos, debido a los cambios imprevistos que se realizaron carentes de diseño y planificación.
- El rendimiento del sistema, ya que los técnicos se quejan de que el sistema suele bloquearse o funcionar con grandes retardos.
- El bloqueo que se produce por los accesos en exclusividad a los datos.

1.2. Objetivos

El objetivo principal de este Proyecto Fin de Carrera (en adelante, PFC) es el desarrollo de una aplicación web de gestión interna de la empresa, sólo para empleados, con la gestión de clientes y sus solicitudes, proyectos y facturas. Debe ser funcional, sin tener demasiado en cuenta el aspecto visual, ya que será usada por los técnicos de la empresa y no por los clientes.

Para ello se pretende adaptar un software empresarial existente con licencia de software libre, como base para producir el nuevo sistema, que tiene que servir para:

- Gestionar los clientes y sus contactos con los que tengamos relaciones comerciales.
- Gestionar el catálogo de ensayos y pautas (conjunto de ensayos) que la empresa ofrece.
- Gestionar presupuestos.
- Gestionar y producir informes con los resultados de los ensayos realizados.
- Gestionar albaranes y facturas para registrar las ventas.

1.3. Glosario y acrónimos

A lo largo del documento se tratarán los siguientes términos con el siguiente significado:

Application Dictionary (AD) Diccionario de la Aplicación: conjunto de metadatos que describen el comportamiento de la aplicación utilizando las propias estructuras de datos de la misma aplicación.

Application Programming Interface (API) Interfaz de Programación de Aplicaciones: es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usadas generalmente en las bibliotecas (también denominadas comúnmente *librerías*).

Commit Entrega de código al repositorio de código del sistema de control de versiones.

Convention over Configuration (CoC) (Convención sobre Configuración). Dícese cuando se intenta establecer las mejores prácticas de programación *por defecto*, de forma que no se necesiten (a veces, incluso sea imposible) personalizar las herramientas que tratan con el código, como ocurre el sistema Maven. Resta flexibilidad pero facilita la estandarización.

Cookie Variable manejada por servidor y cliente (navegador) que permite a la aplicación web reconocer una sesión válida o inválida como herramienta para conseguir la persistencia del estado interno.

Create Implica la creación de un objeto, pudiendo completar todos sus atributos en un formulario de entrada de datos, y su posterior guardado de manera persistente.

Create, Retrieve, Update & Delete (CRUD) Se usa para indicar que se aplica igual para estas cuatro operaciones sobre un tipo de objeto de negocio: Create, Retrieve, Update & Delete. Usaremos CRU para excluir el borrado. En el contexto de los casos de uso, emplearemos «Gestionar X» para unificar los 5 casos de uso básicos sobre la clase de objetos X: Crear X, Modificar X, Listar X, Buscar X, Borrar X [13].

Customer Relationship Mangement (CRM) Software para la Administración de la Relación con los Clientes. Sistemas informáticos de apoyo a la gestión de las relaciones con los clientes, a la venta y al marketing.

Database Management System (DBMS) Los Sistemas de Gestión de Bases de Datos son un tipo de software específico, que sirve de interfaz entre los datos y las aplicaciones que utilizan esos datos [32].

Delete La operación de eliminar persistentemente un objeto conocido. En algunos casos concretos en los que sea inviable o no sea deseable su eliminación, habrá que "tacharlo" o "marcarlo" como borrado (*borrado lógico*) pero mantenerlo en la base de datos.

Ensayo Procedimiento que aplicado a un material u objeto permite extraer información acerca de sus propiedades físicas o químicas.

Enterprise Resource Planing (ERP) Los sistemas de Planificación de Recursos Empresariales son sistemas de gestión de información que integran todo lo necesario para el funcionamiento de los procesos de negocio de la empresa. [32].

Hibernate Query Language (HQL) Abstracción del lenguaje de consulta SQL que proporciona Hibernate para la consulta de objetos.

HyperText Markup Language (HTML) (Lenguaje de Marcado de HiperTexto). Es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes [32].

Integrated Development Environment (IDE) Se trata de una aplicación que integra diversas herramientas de programación para uno o más lenguajes: como un editor, compilador y depurador.

JavaServer Faces (JSF) Framework de desarrollo MVC para J2EE.

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Esta tecnología es un desarrollo de la compañía Sun Microsystems. Las JSP's permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas [32].

Model-driven Development (MDD) es un paradigma de desarrollo de software que se centra en la creación y explotación de modelos de dominio (es decir, representaciones abstractas de los conocimientos y actividades que rigen un dominio de aplicación particular), más que en conceptos informáticos (o algoritmos).

Mozilla Public License (MPL) es una licencia de software libre y abierto, inicialmente creada para Netscape.

Model-View-Controller (MVC) Patrón de diseño que separa la lógica de negocio y los datos, de la interfaz de usuario.

Maven (mvn) Sistema de gestión del código fuente, ideal para proyectos en tecnología Java, que incluye un gestor de dependencias automático y recompilación del proyecto.

Navegador Programa de cliente responsable de representar e interpretar la interfaz entre la persona usuario del sistema y el sistema (aplicación del servidor web).

Lines Of Code (LOC) (Líneas de Código) es una métrica de software usada para estimar el tamaño del sistema y el esfuerzo necesario para su construcción.

Object-oriented Programming (OOP) o Programación Orientada a Objetos: es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos.

Object-relational Mapping (ORM) o Mapeo Objeto-Relacional, es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia.

Objeto de negocio (O simplemente, *Entidad*) Dícese del tipo de objeto que tiene tanto identidad como entidad propias dentro de la lógica del negocio.

Pauta Procedimiento normalizado por alguna organización que asocian un conjunto de ensayos definidos, y en orden, a unos tipos de materiales que sean de aplicación.

Persistencia Propiedad de los sistemas de información de almacenar ésta de manera invulnerable a caídas del sistema operativo o fallos de alimentación, que permita recuperarse sin perder la consistencia y cohesión de la información que es persistente.

Portable Document Format (PDF) Formato de Documento Portable. Es un formato de almacenamiento de documentos, desarrollado por la empresa Adobe Systems. Este formato es de tipo compuesto, en los que se puede incluir imagen vectorial, mapa de bits y texto, del que se disponen lectores para todas las plataformas.

Retrieve La operación de poder recuperar un objeto y examinar su contenido y atributos asociados. Implica también la opción de buscar objetos dado(s) alguno(s) de los valores de búsqueda para cada atributo, así como mostrar e imprimir el listado resultante.

RedHat Package Manager (RPM) Sistema de gestión de paquetería original de la distribución de Linux RedHat y sus derivadas; ampliamente usado.

Scope (Ámbito) En JSF, es el tiempo de vida de cada instancia de cada Bean (controlador). Puede ser `application` (significa que la instancia es compartida entre todos los hilos de ejecución y desde cualquier punto de la aplicación), `session` (una instancia para cada sesión), `request` (una instancia para cada petición: nace y muere con la petición y respuesta del servidor).

Sesión Período de tiempo en el que el usuario de la sesión permanece conectado al sistema asociando una serie de datos temporales necesarios para el buen funcionamiento de la aplicación. Véase *Cookie*.

Servidor de aplicaciones Sistema que vela por la seguridad y la operación correcta de las aplicaciones remotas proporcionando un entorno y marco de trabajo determinado.

Sistema de Gestión de Bases de Datos (SGBD) (O simplemente, *Base de datos*) Sistema que proporciona la persistencia necesaria a las aplicaciones informáticas con las funciones de almacenar y consultar esta información de la manera más conveniente, efectiva, cómoda y ágil.

Structured Query Language (SQL) El lenguaje de consulta estructurado es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar –de una forma sencilla– información de interés de una base de datos, así como también hacer cambios sobre ella.

Software Requirements Specification (SRS) La Especificación de Requisitos del Software es un documento necesario para poder diseñar el sistema software. Es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye la descripción de las funcionalidades que el cliente desea expresados en un lenguaje más formal y sin ambigüedad.

Unified Modeling Language (UML) Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

Update La operación de modificar los atributos de un objeto existente y guardarlo persistentemente.

eXtended Markup Language (XML) Lenguaje estándar de marcado con múltiples aplicaciones para la descripción y/o definición de documentos estructurados.

1.4. Estructura del documento

Este documento está organizado siguiendo el orden cronológico de los procesos implicados durante todo el ciclo de vida. Empezamos con una breve introducción al PFC como primer capítulo. A continuación se explica la planificación (véase capítulo 2) y se describe resumidamente de qué trata el proyecto (véase capítulo 3).

El grueso del PFC comienza con el análisis del sistema, en el capítulo 4, indicando *qué* debe hacer el sistema; seguido de su diseño (véase capítulo 5) definiendo *cómo* debe de hacerlo; e implementación (véase capítulo 6) en la que se muestra la forma de realizarlo. El producto se encuentra implementado en el CD que se adjunta, junto con el código fuente GPLv2.

A continuación se explica cómo se realizaron las pruebas al sistema (véase capítulo 7). Se puede leer un resumen de todo el contenido del PFC en el capítulo 8. Como colofón, concluimos el documento indicando posibles ampliaciones del sistema (véase capítulo 9).

Como apéndices incluimos el manual de instalación (véase apéndice A), el de usuario (véase apéndice B) y el estudio de soluciones alternativas en el apéndice C.

CAPÍTULO 2

Planificación

2.1. Metodología de desarrollo

Para el desarrollo de este Proyecto Fin de Carrera (PFC), hemos seguido la metodología de desarrollo **MÉTRICA versión 3**. MÉTRICA es una metodología de planificación, desarrollo y mantenimiento de sistemas de información promovida por el Ministerio de Administraciones Públicas del Gobierno de España para la sistematización de actividades del ciclo de vida de los proyectos software en el ámbito de las administraciones públicas.

Esta metodología propia está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207 (*Information Technology - Software Life Cycle Processes*) así como en la norma ISO/IEC 15504 SPICE (*Software Process Improvement And Assurance Standards Capability Determination*) [15].

La metodología define una serie de procesos que se pueden completar para el desarrollo del sistema de información, y que encajan con las fases del proyecto que explicamos a continuación en la sección 2.2.

- Planificación de Sistemas de Información (PSI).
- Desarrollo de Sistemas de Información (DSI). Debido a su complejidad, está a su vez dividido en cinco procesos:
  - Estudio de Viabilidad del Sistema (EVS).
  - Análisis del Sistema de Información (ASI).
  - Diseño del Sistema de Información (DSI).
  - Construcción del Sistema de Información (CSI).
  - Implantación y Aceptación del Sistema (IAS).

- Mantenimiento de Sistemas de Información (MSI).

2.2. Calendario

En la figura 2.1 se puede ver un diagrama de *Gantt* con la temporización de las tareas de las que se compone el proyecto. Los tiempos están expresados en días laborables de 7 horas. Sin embargo, en la fase de documentación (Memoria y Presentación) cada día representa una o dos horas de trabajo, ya que estuve trabajando en otros proyectos al mismo tiempo.

- **Entrevistas**: Nos reunimos con el cliente para que nos explique cómo funciona ahora mismo la empresa, y qué cambios son los que necesitan para mejorar su gestión, que son los que tendremos que incorporar a la aplicación. Deberemos consultar a personas que vayan a interactuar con el sistema en distintos roles, para confeccionar una visión global, y objetiva, del producto, resolviendo los posibles conflictos que puedan existir entre ellos. (8 días).
- **Análisis**: Estudiamos las conclusiones extraídas de las entrevistas, y comparamos algunas de las posibles soluciones existentes en el mercado. (10 días).
- **Especificación de requisitos**: En esta fase se desarrolla cada uno de los requisitos del sistema, organizándolos, agrupándolos y describiéndolos en los casos de uso. En particular, es importante determinar si existen requisitos duplicados, parecidos o contradictorios, para resolverlos cuanto antes. (15 días).
- **Diseño**: Durante la fase de diseño, tenemos que estudiar de qué forma vamos a incorporar el comportamiento requerido del sistema a la aplicación, teniendo en cuenta las ventajas de la orientación a objetos, como son la reutilización y el encapsulamiento. También hay que definir cómo vamos a organizar los datos que el sistema necesita manejar. (30 días).
- **Implementación**: Aquí codificaremos lo que sea necesario para que el sistema se comporte como se ha especificado en la etapa de diseño. Incluímos en esta fase la creación de estructuras de datos necesarias en el Sistema de Gestión de Bases de Datos (SGBD). (60 días).
- **Pruebas**: El paso siguiente a la codificación es comprobar que el sistema implementado cumple con la especificación del sistema especificado en las etapas anteriores. Si hubiera algún fallo, es el momento de corregirlo y repetir las pruebas. (20 días).
- **Construcción del prototipo**: Se construye un paquete de software instalable y se rellena con algunos datos iniciales de ejemplo, y una configuración básica. (5 días).
- **Validación del prototipo**: En esta tarea le enseñamos al cliente el producto en funcionamiento, dejando que lo pruebe y que compruebe que cumple con todo lo que acordamos. (5 días).
- **Instalación en cliente**: Instalamos la aplicación integrándolo con su infraestructura dándole una pequeña formación de mantenimiento y monitorización. (5 días).

- **Memoria**: Los documentos que conformarán esta memoria es resultado de detallar los documentos que se han ido obteniendo en todas las anteriores fases del proyecto. (165 días). La memoria del PFC la inicié más tarde, y con menor intensidad de lo esperado por motivos laborales.
- **Pruebas automáticas**: Añadí posteriormente la programación de las pruebas de software con Selenium IDE. (20 días).
- **Presentación**: Como punto final, preparamos la presentación y realizamos la defensa del proyecto frente al tribunal. (15 días).

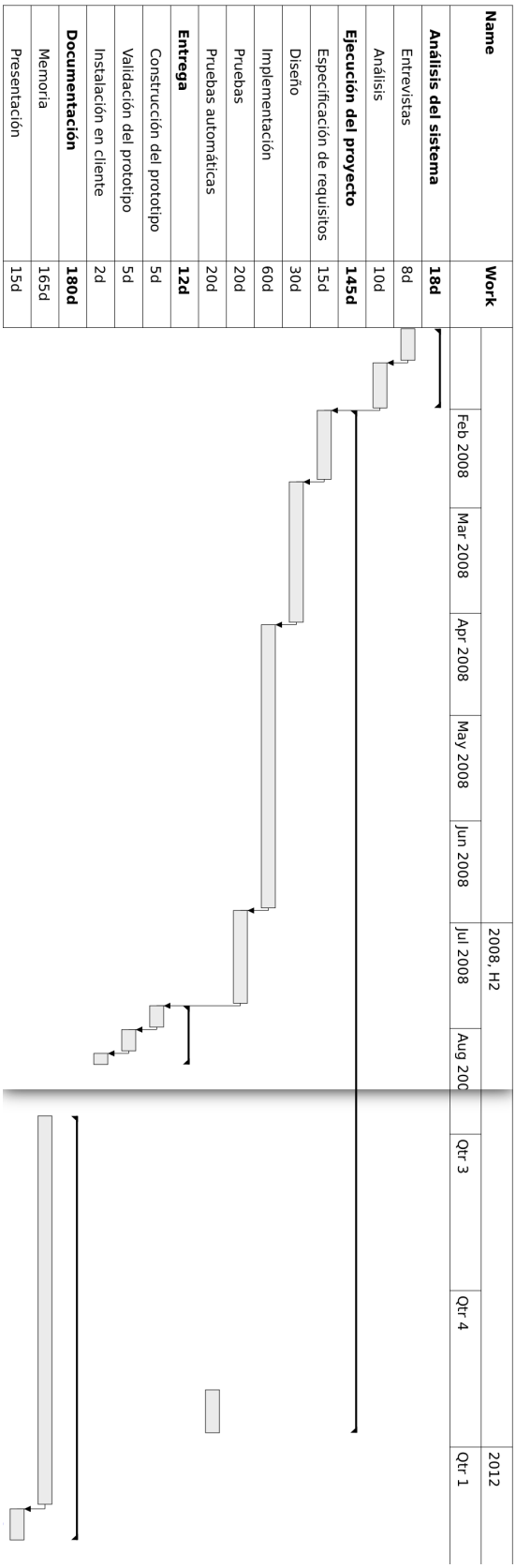

Figura 2.1: Diagrama de Gantt

CAPÍTULO 3

Descripción general del proyecto

En este capítulo explicamos qué se pretende conseguir con la realización de este proyecto, y qué características, desde una perspectiva general, debe cumplir.

3.1. Perspectiva del producto

El sistema software servirá para aumentar la productividad de la empresa al automatizar algunos de sus procesos. Empezaremos con una aplicación de software libre, para después ir adaptándola incorporando todas aquellas funciones que demanden los futuros usuarios. Y para ello, añadiremos componentes nuevos o modificaremos el comportamiento de los existentes que se puedan reutilizar.

3.2. Funciones

En términos generales, el producto software servirá para:

- Gestionar clientes y contactos.
- Realizar facturas y presupuestos.
- Mantener un catálogo de ensayos con sus propiedades,
- y un catálogo de pautas (agrupaciones de ensayos),
- ambos con posibles listas de precios particulares para cada cliente.

- Gestionar a los propios usuarios del sistema (los empleados).
- Obtener informes con los ensayos de cada proyecto para impresión en PDF.

En la sección 4.6 se completa la información relativa a las funciones deseadas del sistema.

3.3. Características del usuario

Los usuarios son los propios empleados de la empresa. Simplemente teniendo conocimientos básicos de informática, y del uso de un navegador, deben ser capaces de realizar sus tareas empleando para ello el sistema software. En la sección 4.1 describimos de forma más detallada los usuarios del sistema.

3.4. Restricciones generales

La aplicación, con arquitectura de cliente/servidor, se podrá consultar vía web desde un navegador. Tiene que funcionar en Windows 2003 Server, y se podrán hacer copias de seguridad periódicas de los datos. En la sección 4.2 se amplía la especificación de restricciones generales.

CAPÍTULO 4

Análisis del sistema

Como en todo proyecto, la primera fase es la elicitación de requisitos. Para ello, se realizarán entrevistas con los futuros usuarios de la aplicación, para poder obtener el siguiente documento, conocido técnicamente como *Especificación de Requisitos del Software* Software Requirements Specification (SRS).

4.1. Características de los usuarios

Los usuarios audiencia del sistema que se pretende construir deben tener conocimientos básicos de la informática de escritorio y del funcionamiento de Internet y los exploradores web. Además, es deseable que posean conocimientos acerca de la interconectividad en redes locales, sistemas Windows® de escritorio y de servidor, trabajo con imágenes, ofimática.

También tendrán que conocer el funcionamiento de la empresa, el flujo de información y de trabajo que siguen, para llevar a cabo registros de todo su trabajo que se reflejen en la aplicación de forma adecuada.

A continuación definimos todos y cada uno de los actores y sus funciones dentro del sistema.

ACT-1	**Usuario**
Actores asociados	
Descripción	Este actor representa a cualquier persona que posee usuario y contraseña, se encuentra registrado en la tabla de usuarios reconocidos del sistema y que puede interactuar con el mismo.
Comentarios	

ACT-2	**Administrador**
Actores asociados	ACT-1
Descripción	Este actor es una especialización del actor **Usuario** (ACT-1). Representa a las personas que poseen privilegios superiores para interaccionar con todos los elementos en su totalidad y sin restricciones, y sobre otros Usuarios.
Comentarios	

ACT-3	**Técnico**
Actores asociados	ACT-1
Descripción	Este actor es una especialización del actor **Usuario** (ACT-1). Representa a los empleados de la empresa, operarios técnicos en su mayoría, de ahí el nombre, con privilegios para usar la aplicación normalmente.
Comentarios	

4.2. Restricciones

Se han identificado las siguientes restricciones y limitaciones de la aplicación:

a) Limitaciones de hardware: El sistema debe poder funcionar en un sistema servidor Dual Xeon 1.6 GHz con 2 GiB de RAM y Windows® 2003 Server.

b) Auditoría: El sistema debe permitir la exportación en un formato libre y estándar de todos los datos bajo demanda, así como llevar un registro de todas las operaciones que efectúa cada usuario sobre cada objeto de negocio.

c) Disponibilidad: Se deben servir sin demasiada latencia las páginas web de la aplicación a una media de 10 usuarios simultáneos de la red local.

d) Protección: Se permitirá habilitar y deshabilitar funciones y partes de la aplicación a distintos roles de usuarios.

e) Seguridad: Establecer un método que posibilite el archivado de toda la información de la base de datos y los archivos asociados.

4.3. Requisitos de interfaces externas

4.3.1. Interfaz con el usuario

Los usuarios del sistema a desarrollar podrán interactuar con éste a través de una interfaz vía web, en un entorno cliente-servidor. Cada usuario debe visualizar las distintas páginas que sirven como interfaz gráfica de la aplicación en un navegador o explorador web instalado en una computadora funcionando bajo un sistema operativo. Se debe asegurar la máxima compatibilidad posible, tanto con los estándares globalmente aceptados, así como con el navegador Internet Explorer 6 en Windows® XP o posteriores versiones.

Las páginas de la aplicación mostrarán un estilo común y una disposición esquemática, intuitiva, orientado a la usabilidad y a la rápida introducción y gestión de los datos por parte de los técnicos.

Deben existir formularios de introducción de datos, así como listados de selección con sus correspondientes formularios de búsqueda. Se necesitará la generación y reproducción de informes y análisis de estadísticas que se crearán a partir de los datos actuales, teniendo en cuenta los parámetros del informe que sean especificados por el usuario, obteniendo como resultado un archivo en formato PDF que se descargue desde el navegador, para ver y/o imprimir.

4.3.2. Interfaz con el hardware

Se debe tener en cuenta que las copias de seguridad que se van a extraer regularmente se realizarán en el mismo servidor que tiene una unidad de disco magneto-óptico (de hasta 250 MB) sobre la que se desean realizar éstas.

4.3.3. Interfaz de software

El software que se desea desarrollar debe funcionar en un entorno Windows® 2003 Server (la parte servidor), mientras que la vista de la parte cliente será interpretada por el navegador web Internet Explorer 6 en Windows XP o superior.

4.4. Objetivos

A continuación se enumeran y detallan los objetivos detectados en la toma de requerimientos que el aplicativo debe contemplar.

OBJ-1	**Gestión de usuarios**
Descripción	El sistema deberá gestionar los distintos usuarios reconocidos por la aplicación, permitiendo a los usuarios con el perfil de Administrador realizar cambios en el conjunto de usuarios, añadiendo nuevos o modificando usuarios y asignando sus roles y conjuntos de permisos. Se incluyen bajo este apartado las funciones de conectarse (*login*) usando nombre y contraseña y desconectarse (*logout*) de la aplicación.
Estabilidad	Alta
Comentarios	

OBJ-2	**Gestión de clientes**
Descripción	El sistema deberá gestionar y cuidar los datos de las organizaciones clientes y personas de contacto, permitiendo su mantenimiento, búsqueda, impresión, etc.
Estabilidad	Alta
Comentarios	

OBJ-3	**Gestión de productos**
Descripción	El sistema deberá mantener un catálogo completo y exahustivo incluyendo los precios de las Pautas y Ensayos que la empresa ofrece. Se podrá imprimir una Pauta en blanco de ejemplo. Debe haber una tabla general de precios y, posiblemente, una tabla particular para cada cliente con precios diferentes (que se puedan actualizar en lote para aplicar un descuento o incremento) y gestionar todas estas tablas. También se pueden gestionar productos que ofrecen los proveedores con su precio de compra.
Estabilidad	Media
Comentarios	

OBJ-4	**Gestión de trabajos**
Descripción	El sistema deberá gestionar adecuadamente Presupuestos, Informes, Albaranes y Facturas, asegurando la trazabilidad entre cada uno de los pasos del flujo completo de trabajo de la empresa, con la posibilidad de obtener un resultado impreso de todos ellos.
Estabilidad	Alta
Comentarios	

4.5. Requisitos de información

En esta sección tratamos los tipos de entidades, relaciones y atributos objeto del dominio del problema que pretendemos resolver con la construcción de este sistema, para satisfacer las necesidades del cliente.

IRQ-1	**Usuarios**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a sus Usuarios. En concreto:
Datos Específicos	▪ Nombre y apellidos ▪ Dirección completa ▪ Usuario y contraseña ▪ Fecha de nacimiento ▪ Rol activo ▪ Departamento al que pertenence
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

IRQ-2	**Departamentos**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a los departamentos que componen la empresa. En concreto:
Datos Específicos	▪ Nombre del departamento ▪ Descripción ▪ Departamento padre que lo contiene, en su caso
Frecuencia	Baja
Estabilidad	Alta
Comentarios	Los departamentos se organizan jerarquizados como un sistema de ficheros. Se representan así «Departamento Padre / Dpto. intermedio / Dpto. intermedio / Departamento» en lo que se conoce como una *línea de trabajo*

IRQ-3	**Clientes**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a los Clientes. En concreto:
Datos Específicos	▪ Nombre, Acrónimo[1] y CIF ▪ Dirección postal completa ▪ Dirección postal de envío de facturas ▪ Dirección postal de envío de informes ▪ Teléfonos, email y página web. ▪ Notas ▪ Si posee Certificación o no, y su estado
Frecuencia	Media
Estabilidad	Alta
Comentarios	

[1]Un *acrónimo* son las siglas, o un código breve de letras para usar de prefijos en sus informes, sus pedidos, etc.

IRQ-4	**Contactos**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a los Contactos conocidos de las empresas cliente. En concreto:
Datos Específicos	▪ Empresa a la que pertenece y su cargo dentro de la misma ▪ Nombre completo ▪ Dirección de email y teléfonos ▪ Si se le ha notificado o no de la inclusión de sus datos al fichero, para complir con la normativa
Frecuencia	Media
Estabilidad	Alta
Comentarios	

IRQ-5	**Ensayos**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a los Ensayos que se realizan en este laboratorio. En concreto:
Datos Específicos	▪ Nombres en inglés y en español del ensayo ▪ Indicador si el ensayo es dimensional o no ▪ Descripción del ensayo ▪ Precio normal (sin I.V.A.) del ensayo ▪ Conjunto de procedimientos aplicables a este ensayo
Frecuencia	Alta
Estabilidad	Alta
Comentarios	

IRQ-6	**Procedimientos**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a los Procedimientos aplicables a cada ensayo. En concreto:
Datos Específicos	▪ Ensayo asociado ▪ Denominación del procedimiento
Frecuencia	Alta
Estabilidad	Alta
Comentarios	

IRQ-7	**Pautas**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	El sistema deberá almacenar la información correspondiente a las Pautas que se realizan en este laboratorio. En concreto:
Datos Específicos	▪ Nombre de la pauta y descripción ▪ Título del informe que se generará ▪ Familia[2] de pautas ▪ Tiempo de respuesta previsto en días. ▪ Material necesitado para el ensayo y agrupación. ▪ Identificadores de la pauta. ▪ Organización que la define, si la pauta es exclusiva de un cliente. ▪ Conjunto de procedimientos y ensayos que componen la pauta. ▪ Precio normal (sin I.V.A.) de la pauta.
Frecuencia	Alta
Estabilidad	Alta
Comentarios	

4.6. Requisitos funcionales

A continuación se especifican los requisitos y casos de uso del sistema.

[2]La *familia* es un clasificador (opcional) de pautas.

4.6.1. Gestión de usuarios

En la figura 4.1 se muestra la relación de los casos de uso relacionados con la gestión de usuarios del sistema. A continuación se describe cada uno de ellos.

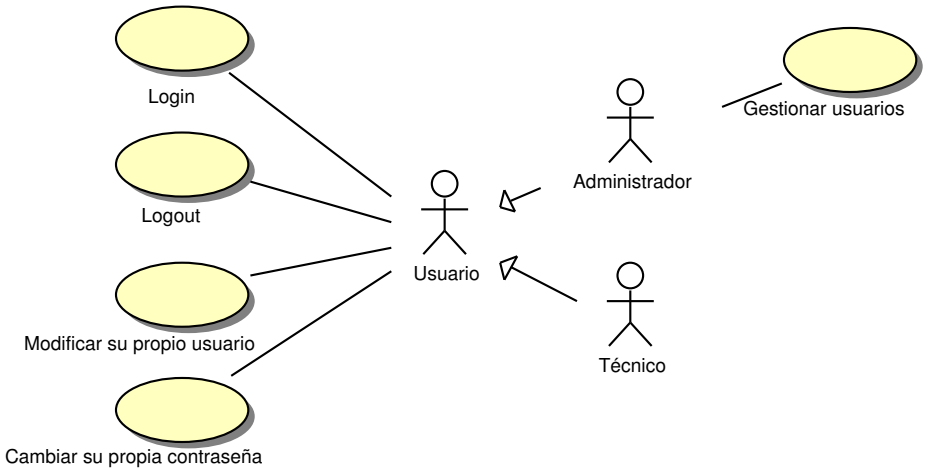

Figura 4.1: Casos de Uso: Gestión de Usuarios

UC-1	**Modificar usuario**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un administrador modificar la información de otro usuario del sistema.
Precondiciones	El usuario seleccionado para modificar existe y el usuario conectado al sistema es un Administrador.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del usuario seleccionado y sus valores: nombre, apellidos, dirección, email, dni, número de la seguridad social y rol. 2. El usuario conectado modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El usuario seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-2	**Crear usuario**
Objetivos asociados	OBJ-1
Requisitos asociados	UC-1
Descripción	Permite a un administrador crear un usuario nuevo en el sistema para que pueda conectarse.
Precondiciones	El usuario conectado al sistema es un Administrador.
Secuencia Normal	1. El sistema crea un usuario nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-1 tomando como entrada este nuevo usuario en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El usuario se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el Administrador cancela la operación, no se produce ningún cambio en el sistema.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-3	**Listar usuarios**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un administrador examinar la lista de usuarios.
Precondiciones	El usuario conectado al sistema es un Administrador.
Secuencia Normal	1. El sistema muestra un listado paginado de usuarios con el nombre y apellidos de cada uno. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-4	**Buscar usuarios**
Objetivos asociados	OBJ-1
Requisitos asociados	UC-3
Descripción	Permite a un administrador buscar usuarios
Precondiciones	El usuario conectado al sistema es un Administrador.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de usuarios, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-3 filtrando los usuarios que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de usuarios coincidentes.
Excepciones	▪ Si no hay usuarios que coincidan con los criterios de búsqueda, aparece un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-5	**Borrar usuario**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un administrador borrar algún usuario
Precondiciones	El usuario conectado al sistema es un Administrador y el usuario seleccionado para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema establece una marca de *inactivo* al usuario seleccionado para ser borrado y lo guarda. 4. El sistema muestra el listado de usuarios restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el usuario impidiendo que se pueda volver a conectar.
Excepciones	▪ Si el Administrador no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	Se ha determinado realizar un «borrado lógico» del usuario, de forma que no aparezca en el listado de usuarios ni se le permita conectarse a la aplicación, para tener la apariencia que el usuario ha sido borrado. El motivo es ahorrarse realizar la búsqueda de todos los elementos del sistema relacionados con este usuario para verificar que no se dejen asociaciones inválidas.

UC-6	**Modificar su propio usuario**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un usuario modificar sus datos personales dentro del sistema.
Precondiciones	El usuario se encuentra conectado
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del usuario conectado al sistema y sus valores: nombre, apellidos, dirección, email, dni, número de la seguridad social, etc. 2. El usuario modifica los atributos deseados y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El usuario ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	Es un caso de uso distinto al UC-1 ya que en este caso el usuario puede modificar todos sus datos excepto el rol, para que no se pueda producir una promoción de permisos no deseada.

UC-7	**Cambiar su propia contraseña**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un usuario modificar su contraseña de acceso al sistema.
Precondiciones	El usuario se encuentra conectado.
Secuencia Normal	1. El sistema muestra un formulario de edición solicitando la contraseña actual y doblemente la nueva. 2. El usuario completa los tres campos y envía el formulario. 3. El sistema cambia la contraseña del usuario conectado por la nueva contraseña remitida. 4. Finaliza el caso de uso.
Postcondiciones	La contraseña del usuario ha sido modificada y guardada.
Excepciones	▪ Si la contraseña actual con coincide con la que tiene actualmente, o la confirmación de contraseña nueva no coincide con la contraseña nueva, se muestra el error y se repite el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-8	**Login**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un Usuario identificarse al sistema e iniciar sesión.
Precondiciones	El Usuario no se encuentra conectado actualmente.
Secuencia Normal	1. El sistema muestra un formulario con campos de nombre de usuario y contraseña. 2. El usuario rellena los campos con sus datos y envía el formulario. 3. El sistema comprueba la validez del usuario y su contraseña, le crea una sesión, y lo redirige a la página inicial de entrada. 4. Finaliza el caso de uso.
Postcondiciones	Se ha creado la sesión para el usuario identificado y autorizado.
Excepciones	▪ Si el usuario no existe, o la contraseña no concuerda, el sistema informa del error y se reinicia el caso de uso.
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-9	**Logout**
Objetivos asociados	OBJ-1
Requisitos asociados	
Descripción	Permite a un Usuario cerrar su sesión y desconectarse del sistema.
Precondiciones	El Usuario se encuentra conectado actualmente.
Secuencia Normal	1. El sistema limpia la sesión del usuario y descarta todos sus datos temporales que hubiere. 2. El sistema redirige al usuario hacia la pantalla inicial de acceso a la aplicación. 3. Finaliza el caso de uso.
Postcondiciones	Se expirado la sesión del usuario y todos sus datos temporales asociados.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

4.6.2. Gestión de clientes

A continuación, se describen los casos de uso relacionados con la gestión de clientes (figura 4.2).

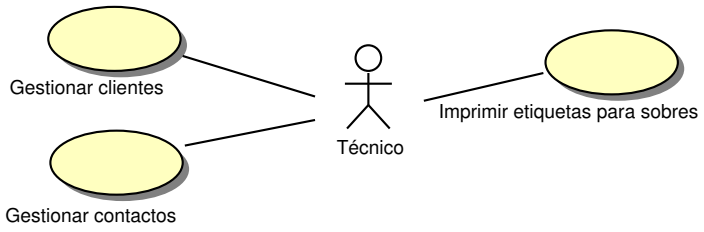

Figura 4.2: Casos de Uso: Gestión de Clientes

UC-10	**Modificar cliente**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de un cliente registrado en el sistema.
Precondiciones	El cliente seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del cliente seleccionado y sus valores: Nombre, CIF, dirección postal, direcciones electrónicas. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El cliente seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-11	**Crear cliente**
Objetivos asociados	OBJ-2
Requisitos asociados	UC-10
Descripción	Permite a un usuario crear un cliente nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un cliente nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-10 tomando como entrada este nuevo cliente en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El cliente se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-12	**Listar clientes**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de clientes.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de clientes con el nombre, teléfono y tipos de cada uno. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-13	**Buscar clientes**
Objetivos asociados	OBJ-2
Requisitos asociados	UC-12
Descripción	Permite a un usuario buscar clientes
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de clientes, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-12 filtrando los clientes que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de clientes coincidentes.
Excepciones	▪ Si no existen clientes que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-14	**Borrar cliente**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario borrar algún cliente
Precondiciones	El usuario se encuentra conectado al sistema y el cliente seleccionado para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el cliente del registro. 4. El sistema muestra el listado de clientes restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado la organización cliente.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-15	**Modificar contacto**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de un contacto registrado en el sistema.
Precondiciones	El contacto seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del contacto seleccionado y sus valores: nombre completo, email, teléfono, empresa y cargo. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El contacto seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-16	**Crear contacto**
Objetivos asociados	OBJ-2
Requisitos asociados	UC-15
Descripción	Permite a un usuario crear un contacto nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un contacto nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-15 tomando como entrada este nuevo contacto en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El contacto se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-17	**Listar contactos**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de contactos.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de contactos con su nombre, teléfono, email y organización a la que pertenece. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-18	**Buscar contactos**
Objetivos asociados	OBJ-2
Requisitos asociados	UC-17
Descripción	Permite a un usuario buscar contactos
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de contactos, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-17 filtrando los contactos que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de contactos coincidentes con las condiciones de búsqueda.
Excepciones	▪ Si no existen contactos que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-19	**Borrar contacto**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un usuario borrar algún contacto de una organización.
Precondiciones	El usuario se encuentra conectado al sistema y el contacto seleccionado para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el contacto del registro de contactos. 4. El sistema muestra el listado de contactos restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el contacto de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-20	**Imprimir etiquetas para sobres**
Objetivos asociados	OBJ-2
Requisitos asociados	
Descripción	Permite a un técnico obtener una página para imprimir etiquetas para los sobres a enviar al cliente.
Precondiciones	El usuario se encuentra conectado.
Secuencia Normal	1. El usuario selecciona la organización de la que quiere obtener la página de etiquetas. 2. El sistema confecciona el informe de etiquetas de envío usando los datos de la dirección postal de la organización seleccionada y se lo devuelve al usuario. 3. El usuario imprime la hoja en un papel de pegatina. 4. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido la hoja de etiquetas.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

4.6.3. Gestión de productos

En la figura 4.3 se muestran los casos de uso relacionados con la gestión de productos, tanto ofertados como necesitados.

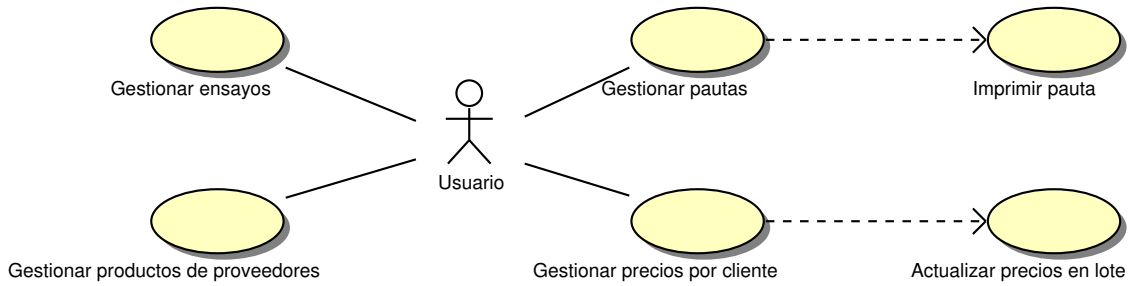

Figura 4.3: Casos de Uso: Gestión de Productos

UC-21	**Modificar ensayo**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de un ensayo registrado en el sistema.
Precondiciones	El ensayo seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del ensayo seleccionado y sus valores: nombre, descripción, lista de procedimientos y coste. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El ensayo seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-22	**Crear ensayo**
Objetivos asociados	OBJ-3
Requisitos asociados	UC-21
Descripción	Permite a un usuario crear un ensayo nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un ensayo nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-21 tomando como entrada este nuevo ensayo en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El ensayo se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-23	**Listar ensayos**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de ensayos.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de ensayos con su nombre y coste normal. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-24	**Buscar ensayos**
Objetivos asociados	OBJ-3
Requisitos asociados	UC-23
Descripción	Permite a un usuario buscar ensayos.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de ensayos, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-23 filtrando los ensayos que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de ensayos coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen ensayos que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-25	**Borrar ensayo**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario borrar un ensayo del registro de productos.
Precondiciones	El usuario se encuentra conectado al sistema y el ensayo seleccionado para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el ensayo del registro de productos. 4. El sistema muestra el listado de ensayos restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el ensayo de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-26	**Modificar pauta**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de una pauta registrada en el sistema.
Precondiciones	La pauta seleccionada para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos de la pauta seleccionada y sus valores: nombre y título, etiquetas, ensayos, etc. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	La pauta seleccionada ha sido modificada y guardada.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-27	**Crear pauta**
Objetivos asociados	OBJ-3
Requisitos asociados	UC-26
Descripción	Permite a un usuario crear una pauta nueva en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea una pauta nueva en blanco en memoria. 2. Se reproduce el caso de uso UC-26 tomando como entrada esta nueva pauta en blanco. 3. Finaliza el caso de uso.
Postcondiciones	La pauta se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-28	**Listar pautas**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de pautas.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de pautas con su nombre y coste normal. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-29	**Buscar pautas**
Objetivos asociados	OBJ-3
Requisitos asociados	UC-28
Descripción	Permite a un usuario buscar pautas.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de pautas, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-28 filtrando las pautas que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de pautas coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen pautas que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-30	**Borrar pauta**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario borrar una pauta del registro de productos.
Precondiciones	El usuario se encuentra conectado al sistema y la pauta seleccionada para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra la pauta del registro de productos. 4. El sistema muestra el listado de pautas restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado la pauta de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-31	**Imprimir pauta**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un técnico obtener un ejemplo de pauta en blanco a partir de los ensayos que lo compondrán.
Precondiciones	El usuario se encuentra conectado y la pauta selecciona existe.
Secuencia Normal	1. El usuario selecciona la pauta para imprimir. 2. El sistema confecciona un informe en blanco a partir de la configuración especificada la pauta y devuelve el PDF resultante. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido el informe en blanco de la pauta.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-32	**Gestionar productos de proveedores**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario gestionar las listas de productos que ofrecen los proveedores[3].
Precondiciones	El Proveedor seleccionado existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra el actual listado de productos del proveedor con su precio asociado. 2. El usuario puede modificar los precios, los nombres de productos, añadir nuevos o borrarlos. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	La tabla de productos y precios ha sido modificada y guardada.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

[3]Un *Proveedor* es igual que un *Cliente* –en definitiva, una *Organización*– pero con la tipología de Proveedor.

UC-33	**Gestionar precios por cliente**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario gestionar precios particulares para los clientes determinados.
Precondiciones	El cliente seleccionado existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra el actual listado de precios especiales para el cliente de los ensayos y pautas. 2. El usuario puede modificar los precios, añadir nuevos ensayos o pautas como de precio especial para este cliente o borrarlos con lo cual se tomaría su precio normal. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	La tabla de precios particulares para el cliente ha sido modificada y guardada.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-34	**Actualizar precios en lote**
Objetivos asociados	OBJ-3
Requisitos asociados	
Descripción	Permite a un usuario actualizar los precios para los clientes de forma masiva.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado de clientes para que el usuario marque aquellos cuyos precios desee modificar. 2. El sistema ofrece la opción de cambiar los precios de todos los Ensayos y/o todas las Pautas para los clientes seleccionados, así como el factor de automultiplicación respecto al precio especial antiguo o el precio normal. 3. El usuario completa el formulario y lo remite. 4. El sistema ejecuta las operaciones indicadas en las tablas de precios. 5. Finaliza el caso de uso.
Postcondiciones	Las tablas de precios particulares se han modificado correctamente en la base de datos.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada (por ejemplo, un factor de automultiplicación negativo) el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

4.6.4. Gestión de trabajos

En la figura 4.4 se muestran los casos de uso relacionados con la gestión de trabajos, núcleo central de la aplicación. Los describimos a continuación.

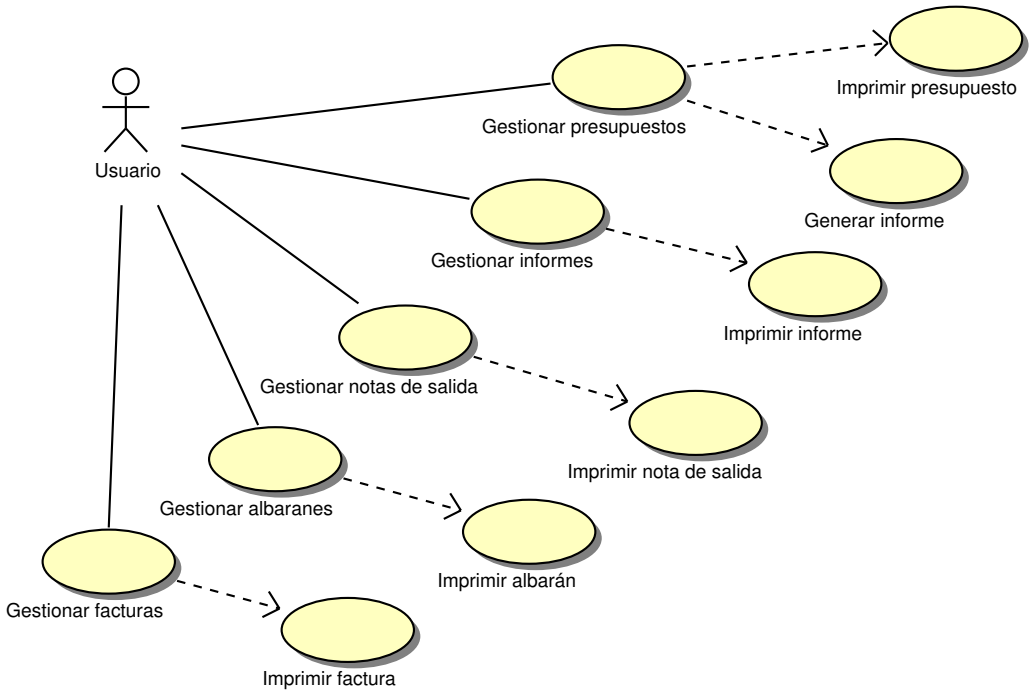

Figura 4.4: Casos de Uso: Gestión de Trabajos

UC-35	**Modificar presupuesto**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario modificar un presupuesto registrado.
Precondiciones	El presupuesto seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del presupuesto seleccionado y sus valores actuales. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El presupuesto seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-36	**Crear presupuesto**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-35
Descripción	Permite a un usuario crear un presupuesto nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un presupuesto nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-35 tomando como entrada este nuevo presupuesto en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El presupuesto se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-37	**Listar presupuestos**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de presupuestos.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de presupuestos con su número de presupuesto, título y cliente. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-38	**Buscar presupuestos**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-37
Descripción	Permite a un usuario buscar presupuestos.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de presupuestos, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-37 filtrando aquellos que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de presupuestos coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen presupuestos que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-39	**Borrar presupuesto**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-37
Descripción	Permite a un usuario borrar un presupuesto del sistema.
Precondiciones	El usuario se encuentra conectado al sistema y el presupuesto seleccionado para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el presupuesto seleccionado. 4. El sistema muestra el listado de presupuestos restantes reproduciendo el caso de uso UC-37. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el presupuesto de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-40	**Imprimir presupuesto**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario obtener un documento imprimible con la información del presupuesto.
Precondiciones	El usuario se encuentra conectado y el presupuesto existe.
Secuencia Normal	1. El usuario selecciona el presupuesto para imprimir. 2. El sistema confecciona un informe con los datos del presupuesto: productos, importes, datos del cliente, fecha, instrucciones y un espacio para escribir observaciones. El sistema devuelve el PDF resultante. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido la vista previa del presupuesto.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-41	**Generar informes**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario crear los informes para el presupuesto en pantalla, como consecuencia de la aceptación del mismo por parte del cliente.
Precondiciones	El usuario se encuentra conectado y el presupuesto existe y todavía no se han creado los informes.
Secuencia Normal	1. El usuario indica el presupuesto del que generar los informes. 2. El sistema crea un informe para cada una de las pautas del presupuesto, y otro informe englobando los costes, tareas y ensayos del presupuesto; prerellenando los datos de cada informe con los de este presupuesto y este cliente, siguiendo las especificaciones de las pautas. 3. El sistema indica las referencias de los informes generados correctamente. 4. Finaliza el caso de uso.
Postcondiciones	Se han creado los informes necesarios en la base de datos.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-42	**Modificar informe**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de un informe registrado en el sistema.
Precondiciones	El informe seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del informe seleccionado y sus valores: identificaciones, descripción, conjunto de tareas, ensayos, costes, ... (ver lista completa de campos en el diagrama 5.7). 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El informe seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-43	**Crear informe**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-42
Descripción	Permite a un usuario crear un informe nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un informe nuevo en blanco en memoria. 2. Se reproduce el caso de uso UC-42 tomando como entrada este nuevo informe en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El informe se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-44	**Listar informes**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de informes.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de informes con las siguientes columnas: cliente, nombre del informe, referencia laboratorio y cliente y fecha de recepción. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-45	**Buscar informes**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-44
Descripción	Permite a un usuario buscar informes.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de informes, intervalo de fechas de recepción y finalización, nombre, descripción, cliente, referencia y estado del informe. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-44 filtrando los informes que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de informes coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen informes que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-46	**Borrar informe**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-44
Descripción	Permite a un usuario borrar un informe registrado.
Precondiciones	El usuario se encuentra conectado al sistema, el informe seleccionado para borrar existe y éste no tiene elementos relacionados, como prespuestos o albaranes.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el informe de la base de datos. 4. El sistema muestra el listado de informes restantes, reproduciendo el caso de uso UC-44. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el informe de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-47	**Imprimir informe**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario obtener el informe de resultados. Hay dos versiones, una para el laboratorio y otra para la empresa.
Precondiciones	El usuario se encuentra conectado y el informe existe.
Secuencia Normal	1. El usuario selecciona el informe y tipo para imprimir. 2. El sistema confecciona el informe particular del tipo seleccionado. El sistema devuelve el PDF resultante. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido la vista previa del informe elegido.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-48	**Modificar albarán**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario modificar la información de un albarán registrado en el sistema.
Precondiciones	El albarán seleccionado para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos del albarán: cliente, número, fecha, si el material ha sido devuelto o no, conjunto de informes incluídos en el albarán, observaciones. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	El albarán seleccionado ha sido modificado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso, sin registrar ningún cambio.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-49	**Crear albarán**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-48
Descripción	Permite a un usuario crear un albarán nuevo en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea un albarán nuevo en blanco en memoria, asignándole el número de albarán que sigue en la secuencia de la forma `A###/YY`, indicando el número secuencial y el año. 2. Se reproduce el caso de uso UC-48 tomando como entrada este nuevo albarán en blanco. 3. Finaliza el caso de uso.
Postcondiciones	El albarán se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-50	**Listar albaranes**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de albaranes.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de albaranes con su referencia y fecha. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-51	**Buscar albaranes**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-50
Descripción	Permite a un usuario buscar albaranes.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de albaranes: referencia, fecha, cliente, informe. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-50 filtrando aquellos que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de albaranes coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen albaranes que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-52	**Borrar albarán**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-50
Descripción	Permite a un usuario borrar un albarán del registro de productos.
Precondiciones	El usuario se encuentra conectado al sistema, el albarán seleccionado para borrar existe y no posee elementos que lo referencien.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra el albarán del registro de productos. 4. El sistema muestra el listado de ensayos restantes reproduciendo el caso de uso UC-50. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado el albarán de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-53	**Imprimir albaran**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario obtener una vista previa del albarán elegido.
Precondiciones	El albarán existe y el usuario de encuentra conectado
Secuencia Normal	1. El usuario pulsa en el botón para crear el informe del albarán. 2. El sistema confecciona el informe con los datos del albarán elegido: información de contacto, referencia, listado de los informes que se envían y devuelve el PDF resultante para descargar. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido el PDF del albarán.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-54	**Modificar nota de salida**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario modificar una nota de salida existente.
Precondiciones	La nota de salida seleccionada para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos de la nota de salida: nombre y fecha, peticionario, cliente original, conjunto de ensayos, ensayos unificados, observaciones. Se incluye un buscador de ensayos según informe que aun estén sin incluir en otras notas de salida, para facilitar la introducción. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos, registrado fecha, hora y usuario que ha realizado la actualización. 4. Finaliza el caso de uso.
Postcondiciones	La nota de salida seleccionada ha sido modificada y guardada.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso, sin registrar ningún cambio.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-55	**Crear nota salida**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-54
Descripción	Permite a un usuario crear una nota de salida nueva en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea una nota de salida nueva en blanco en memoria, asignándole el número de nota de salida que sigue en la secuencia de la forma `NS###/YY`, indicando el número secuencial y el año. 2. Se reproduce el caso de uso UC-54 tomando como entrada esta nueva nota de salida en blanco. 3. Finaliza el caso de uso.
Postcondiciones	La nota de salida se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-56	**Listar notas de salida**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de notas de salida del registro.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de notas de salida con su referencia y fecha. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista.
Excepciones	
Frecuencia	Media
Estabilidad	Alta
Comentarios	

UC-57	**Buscar notas de salida**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-56
Descripción	Permite a un usuario buscar entre todas las notas de salida.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de notas de salida: nombre, rango de fechas, cliente. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-56 filtrando aquellos que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de notas de salida coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen notas de salida que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-58	**Borrar nota de salida**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-56
Descripción	Permite a un usuario borrar una nota de salida del registro de notas de salida.
Precondiciones	El usuario se encuentra conectado al sistema, la nota de salida seleccionada para borrar existe y el sistema no posee elementos que lo referencien.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra la nota de salida del registro de productos. 4. El sistema muestra el listado de notas de salida restantes reproduciendo el caso de uso UC-56. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado la nota de salida de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-59	**Imprimir nota de salida**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario obtener una vista previa de la nota de salida elegida.
Precondiciones	La nota de salida existe y el usuario de encuentra conectado al sistema.
Secuencia Normal	1. El usuario pulsa en el botón para crear el informe de la nota de salida. 2. El sistema confecciona el informe con los datos de la nota de salida elegida: información de contacto, referencia, listado de los ensayos que se envían y devuelve el PDF resultante para descargar. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido el PDF de la nota de salida.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-60	**Modificar factura**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario modificar una factura existente en el sistema.
Precondiciones	La factura seleccionada para modificar existe y el usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario de edición con los atributos de la factura seleccionada y sus valores: número de factura (siguiente autonumérico de la forma `F###/YY`), descripción, número de pedido, línea de trabajo, desglose, albaranes, presupuestos, fecha de emisión, período de facturación, estado. 2. El usuario modifica los atributos necesarios y envía el formulario. 3. El sistema valida la información proporcionada y guarda los datos en la base de datos. 4. Finaliza el caso de uso.
Postcondiciones	La factura seleccionada ha sido modificada y guardada.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-61	**Crear factura**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-60
Descripción	Permite a un usuario crear una factura nueva en el sistema.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema crea una factura nueva en blanco en memoria. 2. Se reproduce el caso de uso UC-60 tomando como entrada esta nueva factura en blanco. 3. Finaliza el caso de uso.
Postcondiciones	La factura se ha creado y guardado.
Excepciones	▪ Si no se ha validado correctamente la información proporcionada el sistema muestra el mensaje de error y se reinicia el caso de uso. ▪ Si el usuario cancela la operación, no se produce ningún cambio en la base de datos.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-62	**Listar facturas**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario examinar la lista de facturas.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un listado paginado de facturas con su número, proveedor/cliente, informe, descripción, fecha, estado, importe total. 2. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista o una lista vacía en caso de no haber facturas.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-63	**Buscar facturas**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-62
Descripción	Permite a un usuario buscar facturas.
Precondiciones	El usuario se encuentra conectado al sistema.
Secuencia Normal	1. El sistema muestra un formulario con los criterios de búsqueda de facturas, con todos los campos en los que buscar valores que coincidan. 2. El usuario completa los campos que desee con los valores de búsqueda aproximada. 3. Se reproduce el caso de uso UC-62 filtrando las facturas que cumplan las condiciones de búsqueda. 4. Finaliza el caso de uso.
Postcondiciones	Se muestra la lista de facturas coincidentes con las opciones de búsqueda.
Excepciones	▪ Si no existen facturas que coincidan con los criterios de búsqueda, se muestra un listado vacío.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-64	**Borrar factura**
Objetivos asociados	OBJ-4
Requisitos asociados	UC-62
Descripción	Permite a un usuario borrar una factura del registro.
Precondiciones	El usuario se encuentra conectado al sistema y la factura seleccionada para borrar existe.
Secuencia Normal	1. El sistema pregunta una confirmación de borrado. 2. El usuario responde que desea borrarlo. 3. El sistema borra la factura del sistema. 4. El sistema muestra el listado de facturas restantes. 5. Finaliza el caso de uso.
Postcondiciones	Se ha borrado la factura de la base de datos.
Excepciones	▪ Si el usuario no confirma el borrado, el sistema no hace nada.
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

UC-65	**Imprimir factura**
Objetivos asociados	OBJ-4
Requisitos asociados	
Descripción	Permite a un usuario obtener una vista previa de la factura elegida.
Precondiciones	La factura existe y el usuario de encuentra conectado.
Secuencia Normal	1. El usuario pulsa en el botón para crear el informe de la factura. 2. El sistema confecciona el informe con los datos de la factura elegida: información de contacto, referencia, desglose importes de los informes y ensayos que se facturan, importe total; y devuelve el PDF resultante para descargar. 3. Finaliza el caso de uso.
Postcondiciones	El usuario ha obtenido el PDF de la factura.
Excepciones	
Frecuencia	Baja
Estabilidad	Alta
Comentarios	

CAPÍTULO 5

Diseño del sistema

Para implementar toda esta funcionalidad que se require en el sistema software, seguiremos diversos estándares J2EE, de forma que queda establecido **Java** como lenguaje de programación a usar.

5.1. Arquitectura

La parte que debemos implementar del software es sólamente un subconjunto de todos los componentes que colaboran entre sí y son el soporte para nuestra aplicación.

Como ayuda en el diseño del sistema cliente/servidor vamos a basarnos en el patrón de diseño *Modelo-Vista-Controlador* (MVC) [8]. Para ello seguiremos la especificación JavaServer Faces (JSF) [30], usando la implentación de Apache MyFaces.

Para aplicar este patrón hay que dividir la funcionalidad en 3 capas:

Modelo Se compone de las clases necesarias para el acceso a datos, búsquedas, lecturas/escrituras, transformaciones de los mismos, etc.

Vista Aquí se agrupan todos los componentes gráficos, visuales, para mostrar, interpretar y gestionar todos los aspectos de la interfaz del usuario.

Controlador Los controladores se encargan de gestionar las peticiones que lleguen al servidor, usando funciones de los modelos que sean necesarios para las distintas vistas.

Para la persistencia de datos, vamos a emplear la biblioteca ampliamente usada *Hibernate* que provee las funciones de un motor de persistencia de tipo ORM (mapeo objeto-relacional), proporcionando la abstracción necesaria para usar una base de datos relacional, como lo es MySQL, como un repositorio organizado de objetos. La parte del *Modelo* del patrón MVC estaría compuesta por esta biblioteca además de los siguientes paquetes:

- `com.autentia.intra.businessobject`: clases de negocio, serializables, con los getters/setters de las propiedades de cada tipo de entidad.
- `com.autentia.intra.dao.hibernate`: clases que se encargan de recuperar, guardar, listar y buscar objetos de negocio empleando para ello funciones de Hibernate (DAO's).
- `com.autentia.intra.dao.search`: clases con métodos para establecer criterios de búsqueda y ordenación, y generar el SQL necesario, para cada clase de negocio.
- `com.autentia.intra.manager`: una abstracción más sobre los DAO's, esta vez gestionados por Spring framework, para tener en cuenta los permisos de acceso a los datos para cada perfil de usuario.

Por otro lado, la *Vista* son las plantillas, los ficheros `*.jsp` que usan etiquetas HTML y también componentes JSF de Apache MyFaces y las extensiones de Apache Tomahawk, así como los recursos necesarios (scripts de JavaScript, hojas de estilo CSS, iconos, ...). En estas plantillas se hacen llamadas a los métodos de los controladores.

Y por último, los *Controladores* están en los paquetes `com.autentia.intra.bean.*`, separados por secciones; haciendo uso de los managers, la biblioteca de JasperReports [10] para generar informes en PDF, y del sistema de navegación entre vista y vista que proporciona el framework JSF.

5.1.1. JavaServer Faces (JSF)

JavaServer Faces es un framework de desarrollo que proporciona las siguientes características [32]:

- Un conjunto de API's para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages (JSP) que permiten expresar una interfaz JavaServer Faces (JSF) dentro de una página JSP.
- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

En JSF, los componentes son etiquetas XML que se traducen en más etiquetas y código en JSP, que el contenedor de aplicaciones (*Tomcat* en este caso) traduce a código Java, lo compila y lo ejecuta para generar código en HTML interpretable por el navegador.

5.1.2. Hibernate

Hibernate es una biblioteca que proporciona los servicios de Object-relational Mapping (ORM). Básicamente mapea cada clase en una tabla de base de datos, y cada atributo de cada clase en una columna en su tabla de la base de datos. En Hibernate se pueden guardar y recuperar objetos, a partir de su clase y su identificador (si estamos guardando un objeto nuevo, el identificador se genera automáticamente incrementando el valor la secuencia en la tabla).

O bien se pueden devolver listados de objetos que cumplan con una serie de criterios de búsqueda, que se pueden especificar en un lenguaje propio, llamado Hibernate Query Language (HQL). Este lenguaje es un *pseudo*-lenguaje, no estándar, muy parecido al lenguaje Structured Query Language (SQL), que este sí es el estándar en bases de datos.

En la versión utilizada de Hibernate, hay que especificar una configuración del mapeo para cada clase, indicando los tipos de los datos, las restricciones, y los atributos que se van a mapear; ya que la clase puede tener otras propiedades que no se desea hacer persistente, como es el caso de propiedades calculadas o derivadas a partir de otra información. Un ejemplo de esto, es el campo `totalAmount` en la clase **Bill**, que es la suma de los conceptos que se están facturando. El valor de la suma, no hay que guardarlo en la base de datos, pues en ese caso no se estaría respetando las reglas de normalización de bases de datos, como la redundancia.

Estas configuraciones se describen en ficheros XML.

Por lo tanto, al emplear estas técnicas, no es necesario diseñar la estructura de la base de datos, ya que es transparente para el programador. Basta con diseñar las clases con sus propiedades y sus relaciones (véanse en la sección 5.2). Hibernate se encargará de traducirlo a tablas y columnas.

5.1.3. Diagrama de componentes y estructura

Podemos ver el diagrama de componentes en la figura 5.1.

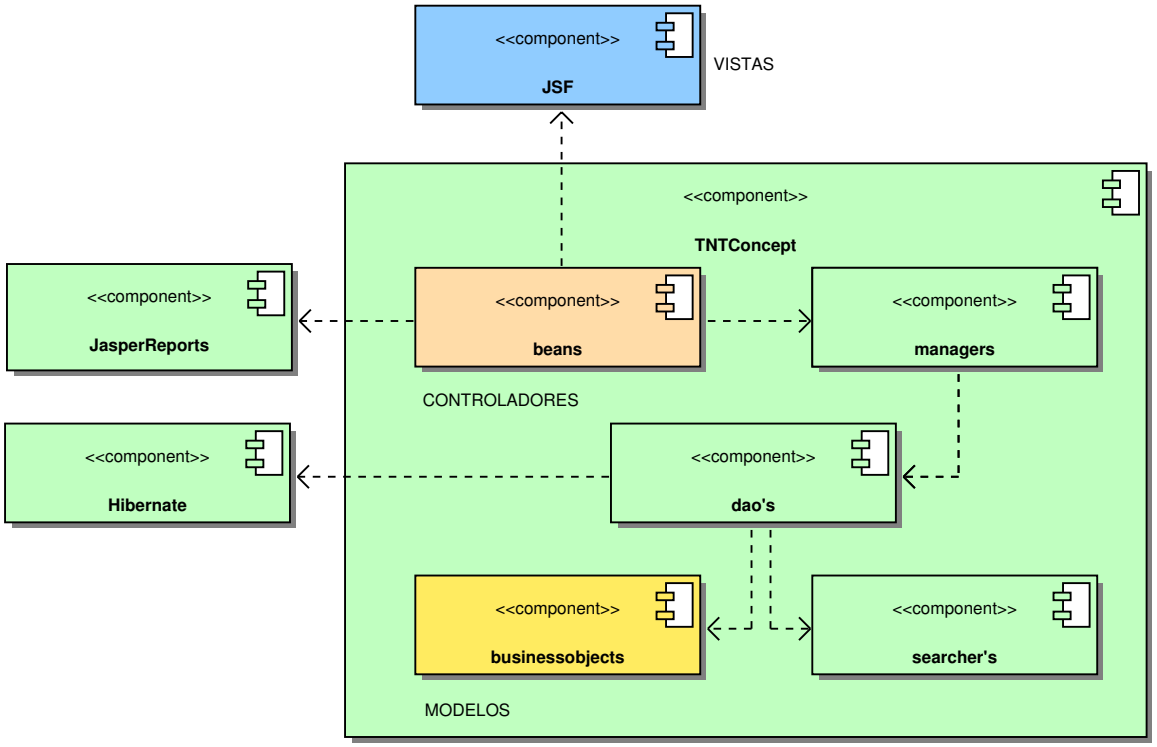

Figura 5.1: Diagrama de componentes software

Al emplear el sistema *Maven* como gestor del código fuente, de la compilación y recompilación, tenemos que ceñirnos a su forma de estructurar el árbol de directorios para los fuentes (siguiendo la filosofía *Convention over Configuration*).

- `/tntconcept/` raíz del proyecto Maven
- `/tntconcept/pom.xml` descriptor del proyecto Maven
- `/tntconcept/src/main` ficheros fuente principales
- `/tntconcept/src/main/bin` raíz para archivos ejecutables `*.sh` con los scripts de *pre* y *post* instalación para la construcción del empaquetado RPM.
- `/tntconcept/src/main/java` raíz de paquetes Java incluyendo ficheros de código fuente `*.java`
- `/tntconcept/src/main/resources` raíz de paquetes Java incluyedo otros tipos de recursos (`*.bmp`, `*.xml`, `*.properties`, ...)
- `/tntconcept/src/main/webapp` raíz de ficheros del contexto web, plantillas `*.jsp`, hojas de estilos `*.css`, imágenes de botones y logos `*.png`, `*.jpg`, `*.gif`, ...
- `/tntconcept/src/test` ficheros fuente de pruebas

- `/tntconcept/target` directorio de salida, con los compilados binarios (no versionar, se generan a partir de los ficheros fuente.)

En el fichero `pom.xml` se definen, entre otras cosas, las dependencias, autogestionadas por Maven, y cuyos *jars* no se necesitan versionar ya que se pueden conseguir de los repositorios, con lo que queda un proyecto más limpio y ordenado.

5.1.4. Entorno de ejecución

A continuación podemos ver el diagrama de despliegue de la puesta en producción (figura 5.2). Interviene un servidor y uno o más clientes, puesto que se ha definido una arquitectura para el sistema de cliente/servidor. El sistema servidor se compone del servidor de aplicaciones J2EE Tomcat 6 y del sistema de gestión de bases de datos MySQL 5. En el servidor de aplicaciones se desplegará un artefacto de aplicación web `tntconcept.war` en el contexto denominado `/tntconcept`, y en el SGBD se creará un usuario y una base de datos para esta aplicación en exclusiva.

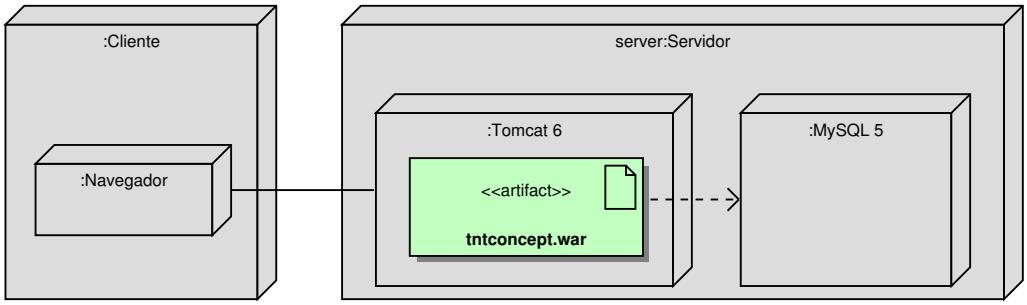

Figura 5.2: Diagrama de despliegue: Entorno de producción

5.2. Diagramas de clases

A partir de los documentos generados durante la toma de requisitos (véase el capítulo 4), y el estudio que se realizó para conocer las particularidades de la herramienta de Autentia TNTConcept, procedemos a definir el modelos de datos.

Los identificadores de clases o de atributos que estén en inglés son elementos que existían en el software original, mientras que aquellos que estén en castellano significa que han tenido que ser añadidos para este PFC.

5.2.1. Usuarios

Podemos ver a continuación un diagrama de las clases del dominio que soportarán la información relativa a la gestión de usuarios, en la figura 5.3. La entidad central es el Usuario, y sus relaciones con el resto de propiedades que pueda tener y otras entidades.

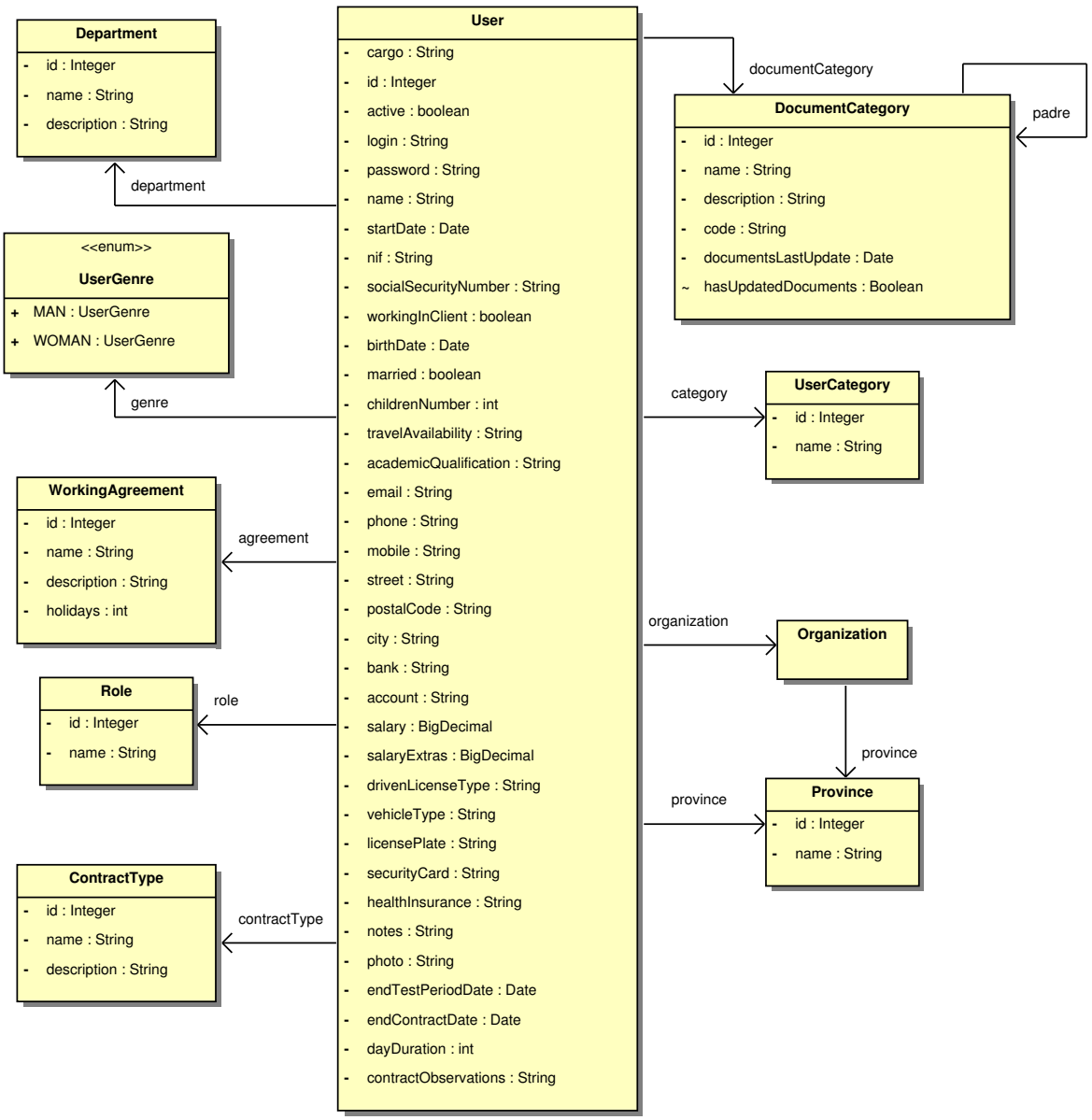

Figura 5.3: Diagrama de clases: Usuarios

5.2.2. Clientes

En la figura 5.4 se puede observar el diagrama de las clases relativas a la información de nuestros clientes y proveedores.

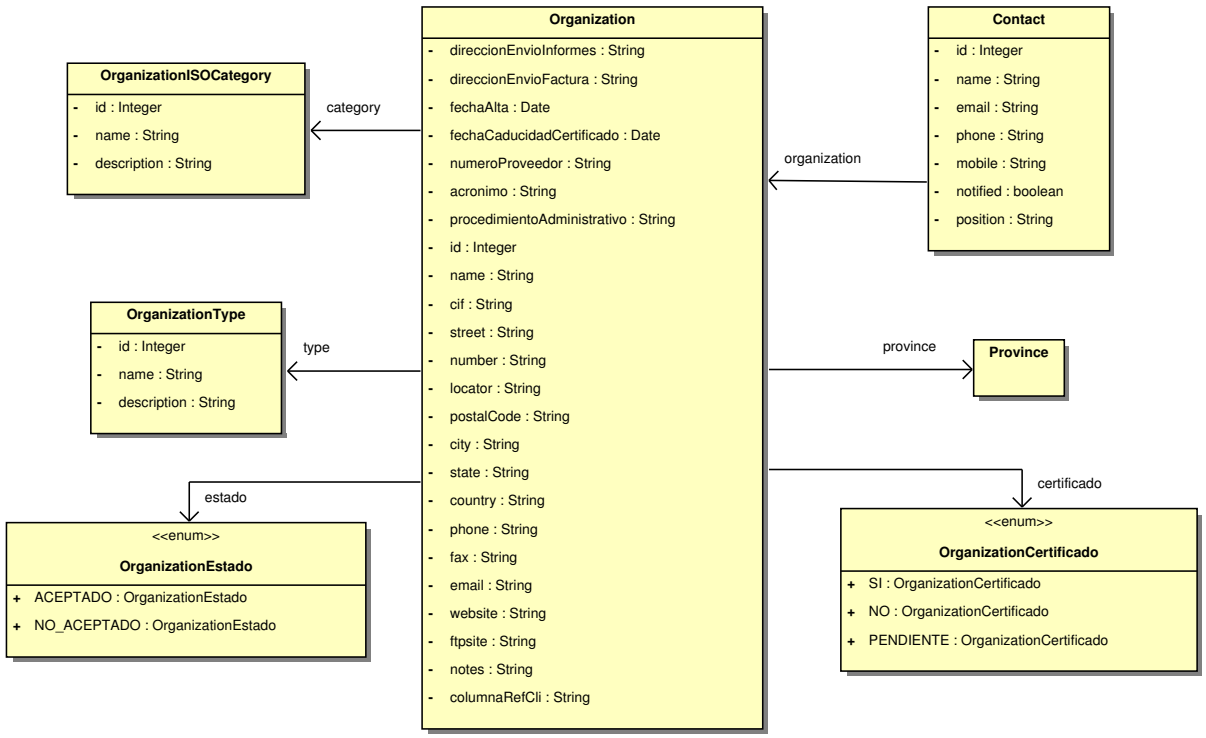

Figura 5.4: Diagrama de clases: Clientes

5.2.3. Productos

A continuación, en la figura 5.5 diseñamos las relaciones entre los modelos que componen el subsistema de Gestión de Productos

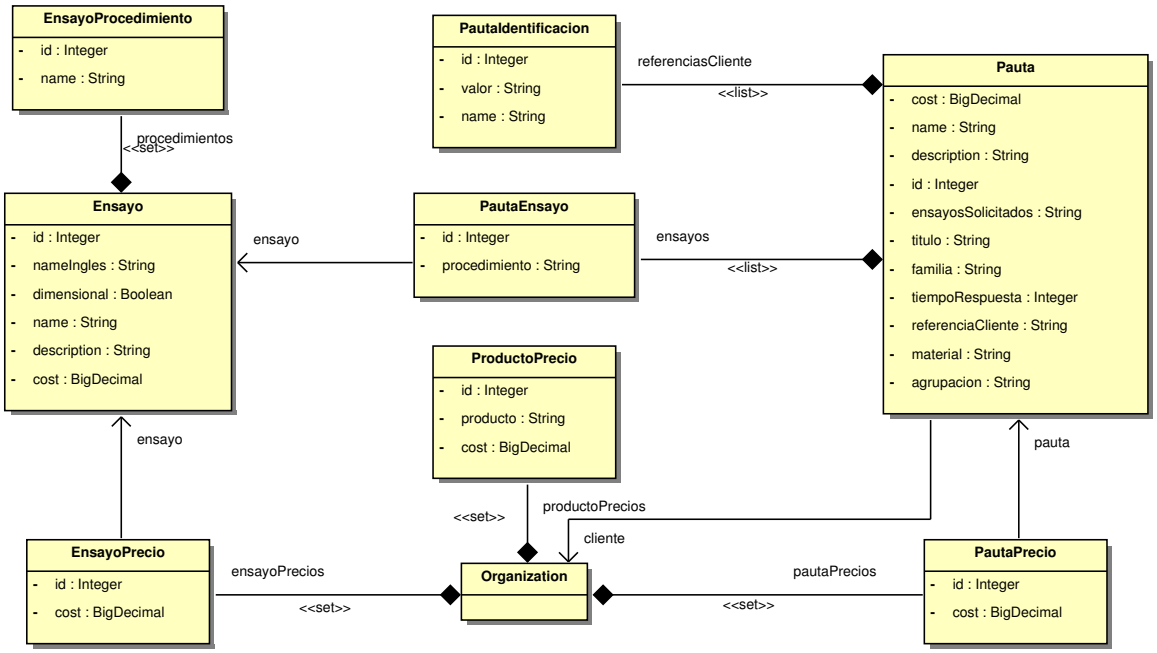

Figura 5.5: Diagrama de clases: Productos

5.2.4. Presupuestos

En la figura 5.6 podemos observar las relaciones entre las entidades con información relativa a los presupuestos.

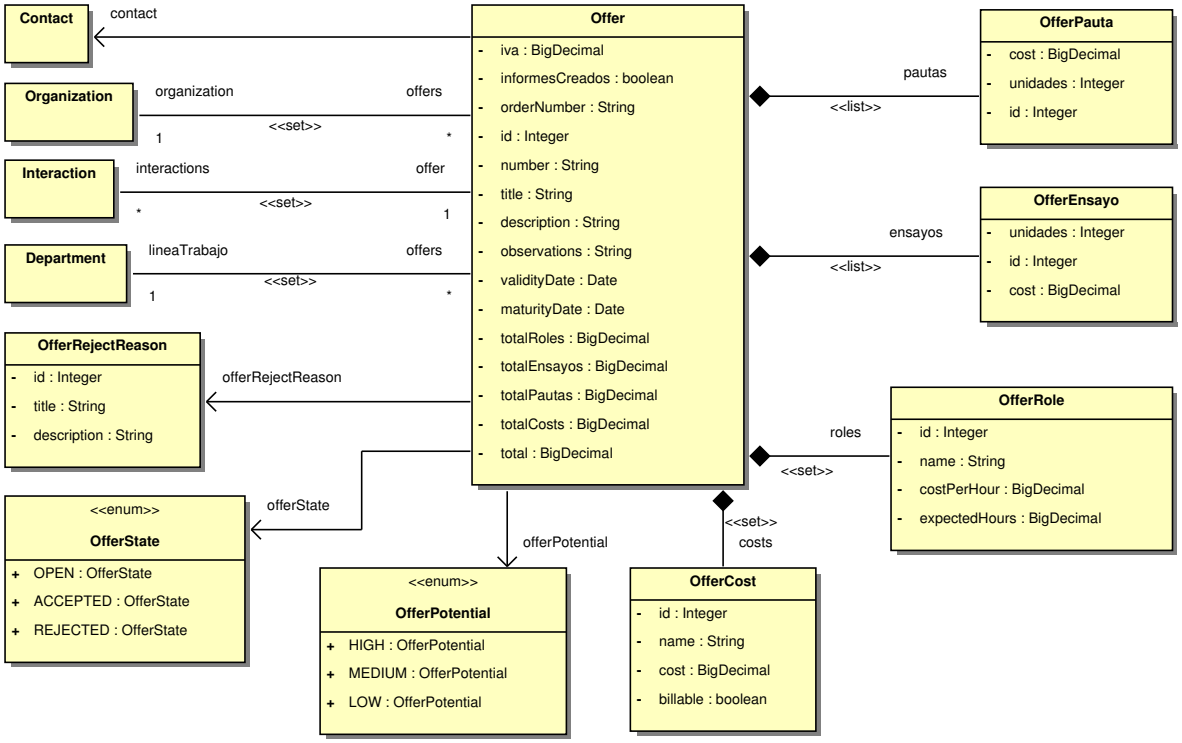

Figura 5.6: Diagrama de clases: Presupuestos

5.2.5. Informes

A continuación, en la figura 5.7, se muestra las relaciones entre la clase Informe y el resto de entidades.

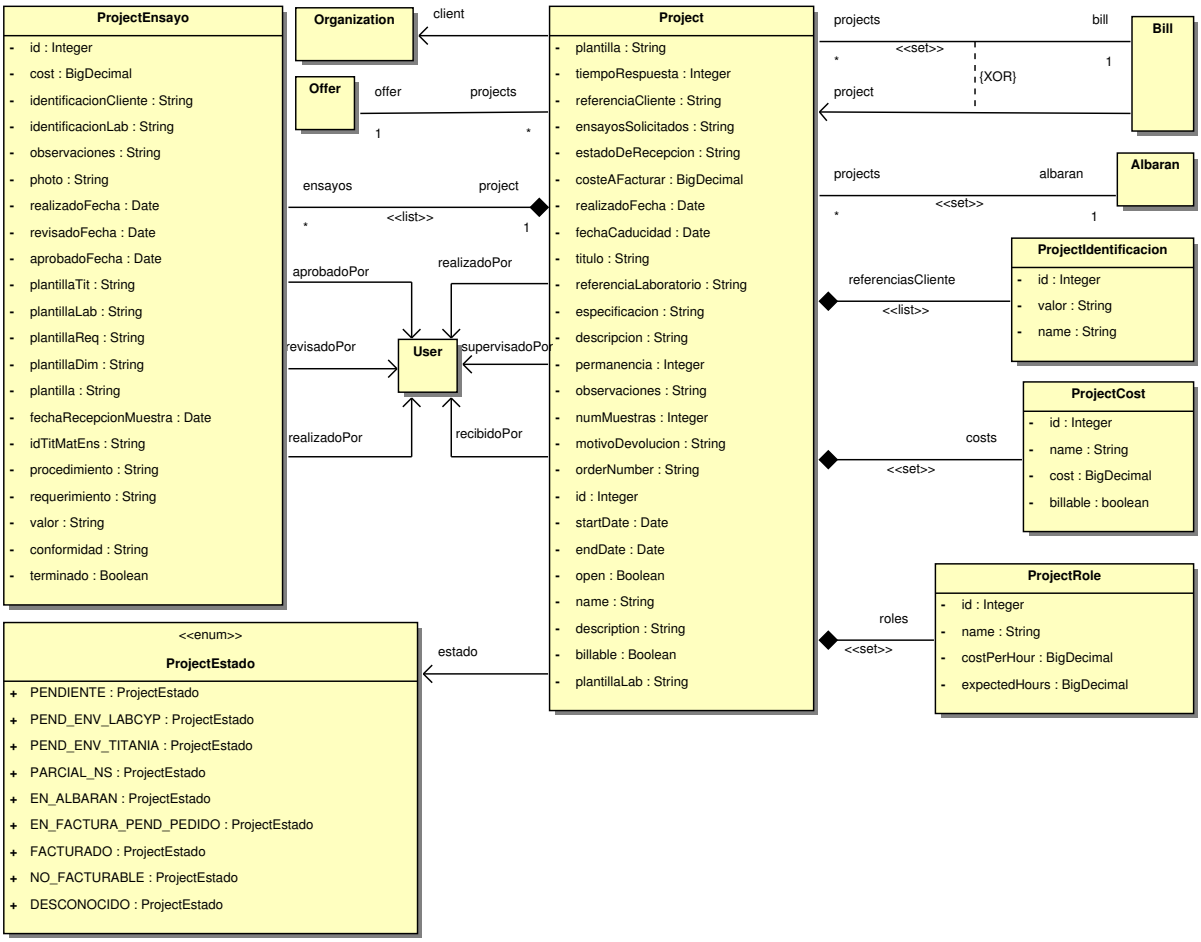

Figura 5.7: Diagrama de clases: Informes

5.2.6. Facturas

En la figura 5.8 se diseña las relaciones entre las clases con la información de las facturas.

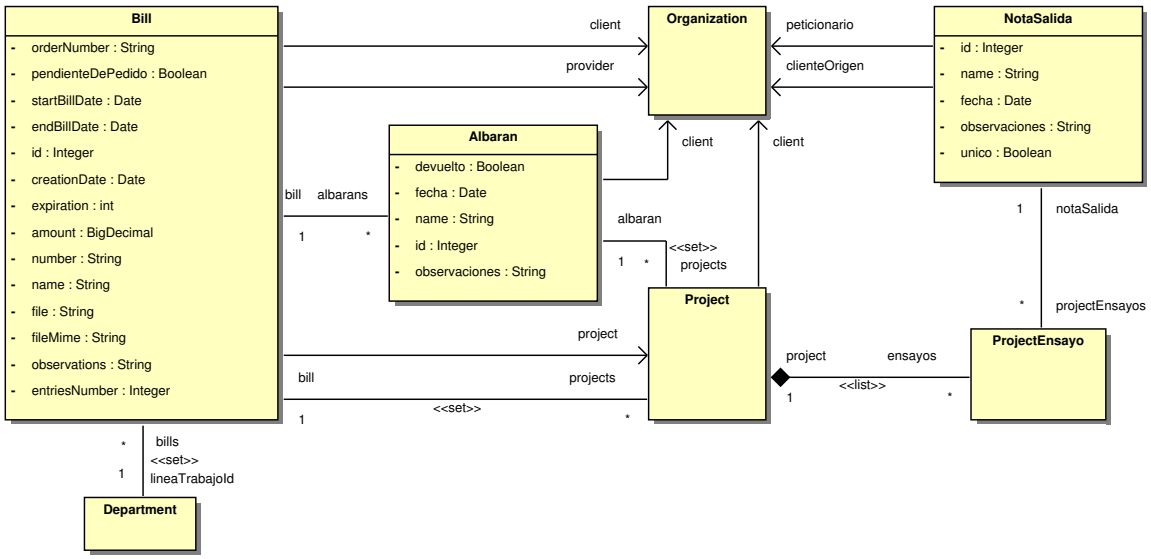

Figura 5.8: Diagrama de clases: Facturas

5.3. Paquetes

A continuación describimos las clases que se contienen en cada paquete, con sus contratos correspondientes.

Recordamos los tipos de propiedades en las clases de tipo JavaBean, con esta leyenda:

1. (R) Clase **fooBar**: (propiedad de sólo lectura) se define el miembro privado `Clase fooBar`, y el método público `Clase getFooBar()` si es de un tipo distinto a Boolean, o bien `Clase isFooBar()` si el tipo es Boolean, devolviendo su valor.
2. (W) Clase **fooBar**: (propiedad de sólo escritura) se define el miembro privado `Clase fooBar`, y el método público `void setFooBar(Clase fooBar)`, que establezca tal valor.
3. (RW) Clase **fooBar**: (propiedad de lectura y escritura) la unión de los dos anteriores.

Además, tienen la obligación de proporcionar un constructor que no acepte argumentos (*nullary constructor*), aunque se permite que existan más constructores.

5.3.1. `com.autentia.intra.bean`

Cada Bean lleva asociado el ámbito JSF que se configura en el fichero `faces-config.xml`. Salvo especificación contraria, el ámbito es *session*.

5.3.1.1. ApplicationBean

Se usa para almacenar variables globales de nivel de aplicación, compartidos para todas las sesiones. Ámbito: *application*.

- (R) String **build** número de versión de la aplicación.

5.3.1.2. ChangePasswordBean

Back-bean para la pantalla de cambio de su propia contraseña. Ámbito: *request*.

- `/pages/admin/changePassword.jsp`

Propiedades y métodos:

- (RW) String **passwordOld** contraseña actual.
- (RW) String **password** contraseña nueva.
- (RW) String **passwordRepe** repetición de la contraseña nueva.
- String **changePassword**() valida las contraseñas, y devuelve el error o confirmación del cambio en la misma página.

5.3.2. `com.autentia.intra.bean.admin`

La sección admin contiene las páginas de administración.

5.3.2.1. UserBean

Back-bean para las pantallas de gestión de usuarios.

- `/pages/admin/detailUser.jsp`
- `/pages/admin/editUser.jsp`
- `/pages/admin/searchUser.jsp`
- `/pages/admin/users.jsp`
- `/pages/admin/userPasswordReset.jsp`

Propiedades y métodos:

- (RW) User **user** usuario de trabajo activo.
- (R) Boolean **userSelected** indica si existe usuario elegido.
- (R) UserSearch **search** objeto con los criterios de búsqueda para usuarios.
- static UserManager **manager** gestor de objetos de tipo `User`.
- String **create**() crea un nuevo usuario en blanco y redirige a la página de edición. Véase 5.9.
- String **delete**() borra el usuario actual y vuelve a la página del listado. Véase 5.12.
- String **detail**() va a la página de edición o de consulta de los datos del usuario elegido. Ver 5.10.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre en los criterios de búsqueda.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.13.
- String **resetPassword**() genera una clave aleatoria que establece para ese usuario, lo guarda y devuelve a la página `userPasswordReset.jsp` con el detalle de la clave nueva.
- String **save**() guarda o actualiza el objeto `User` actual, volviendo a la misma página. Ver 5.11.
- String **search**() redirige al formulario con los criterios de búsqueda de usuarios (`/pages/admin/searchUser.jsp`).

En la figura 5.9 se observa cómo se contruye un usuario nuevo, estableciendo sus campos requeridos con valores por defecto.

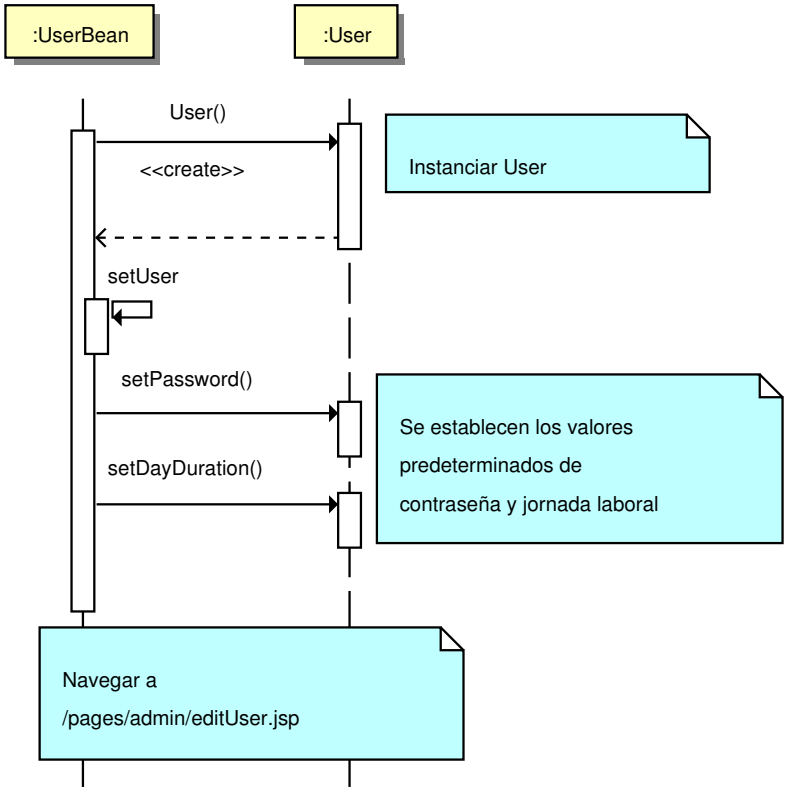

Figura 5.9: Diagrama de secuencia: Crear usuario

En la figura 5.10 se muestran los pasos necesarios para editar la información de un usuario: recuperar el objeto y enviarlo a la vista.

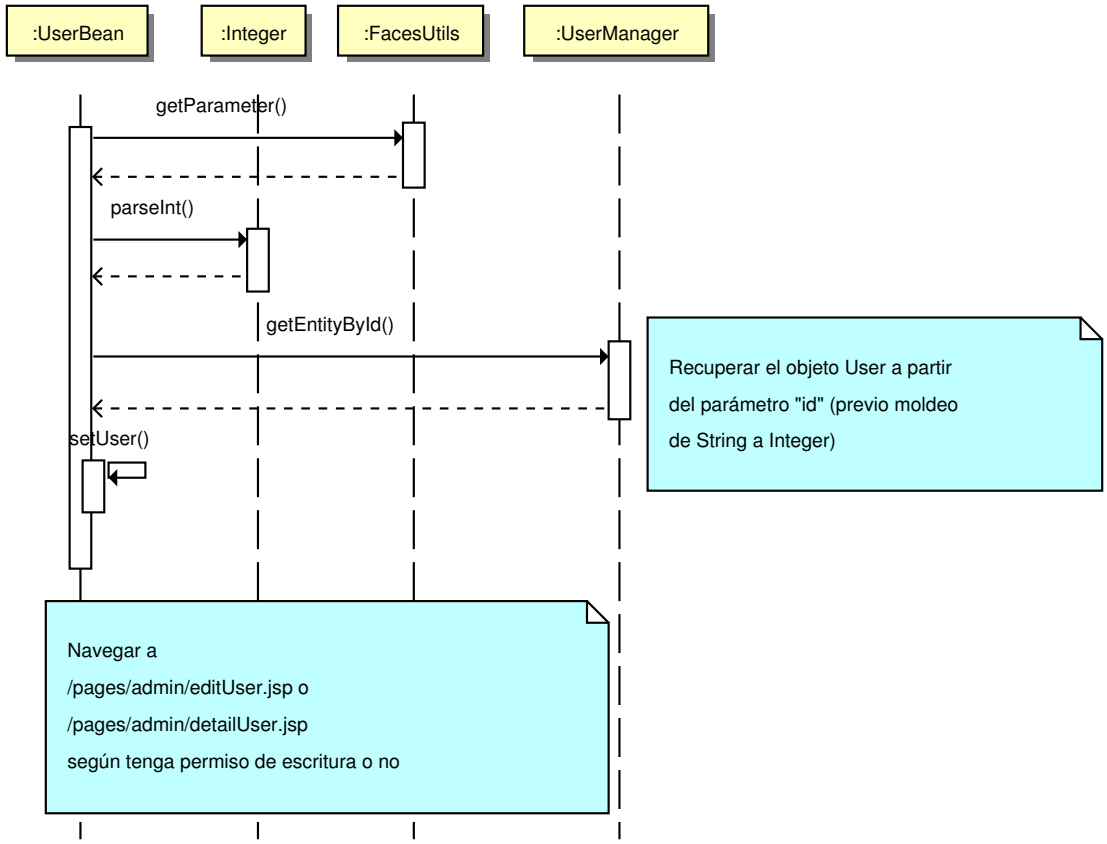

Figura 5.10: Diagrama de secuencia: Editar usuario

En la figura 5.11 podemos ver cómo hay que guardar el usuario, insertando un nuevo objeto o actualizándolo si ya existe.

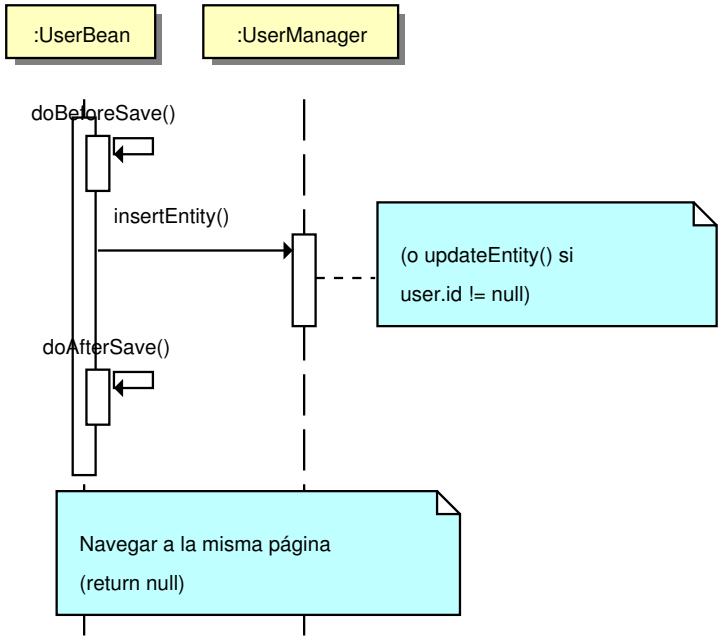

Figura 5.11: Diagrama de secuencia: Guardar usuario

La secuencia necesaria para borrar un usuario se puede ver a continuación, en la figura 5.12

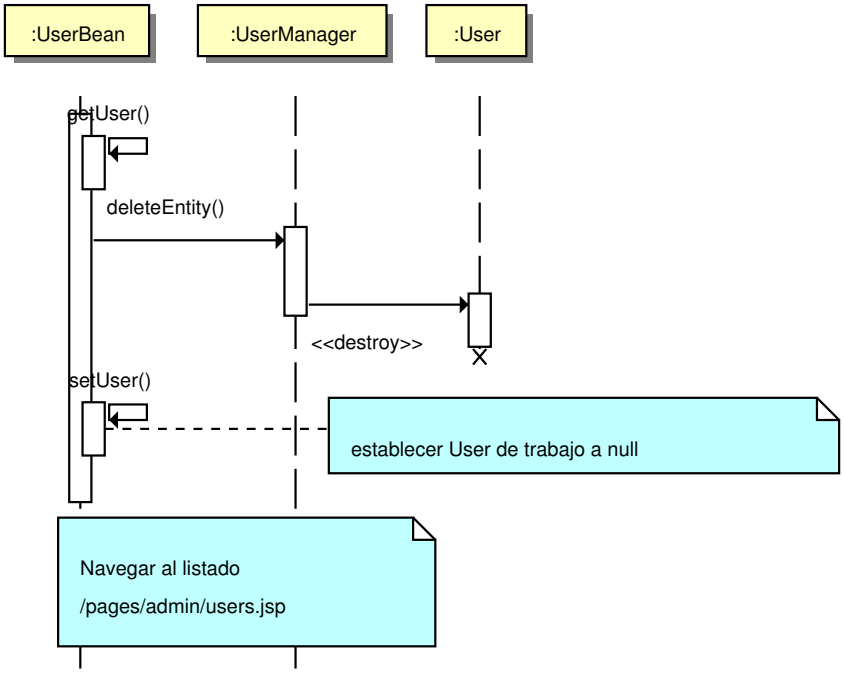

Figura 5.12: Diagrama de secuencia: Borrar usuario

La figura 5.13 muestra la secuencia para eliminar los filtros de búsqueda.

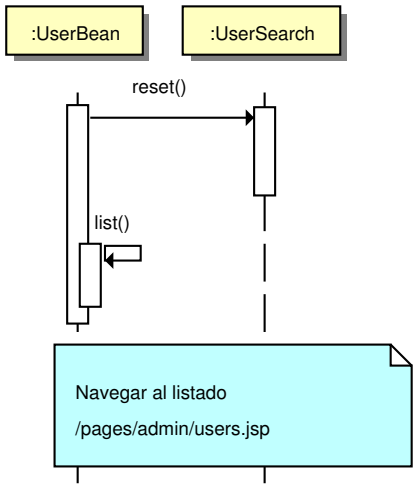

Figura 5.13: Diagrama de secuencia: Resetear búsqueda de usuarios

5.3.3. `com.autentia.intra.bean.contacts`

La sección contacts contiene las páginas de gestión de clientes.

5.3.3.1. OrganizationBean

Back-bean para las pantallas de gestión de usuarios:

- `/pages/contacts/detailOrganization.jsp`
- `/pages/contacts/editOrganization.jsp`
- `/pages/contacts/searchOrganization.jsp`
- `/pages/contacts/organizations.jsp`
- `/pages/contacts/editarPrecios.jsp`

Propiedades y métodos:

- (RW) Organization **organization** organización de trabajo activa.
- (R) Boolean **organizationSelected** indica si existe organización elegida.
- (R) OrganizationSearch **search** objeto con los criterios de búsqueda para organizaciones.
- static OrganizationManager **manager** gestor de objetos de tipo `Organization`.
- String **create**() crea una nueva organización en blanco y redirige a la página de edición. Véase 5.14.
- String **delete**() borra la organización actual y vuelve a la página del listado. Véase 5.17.
- String **detail**() va a la página de edición o de consulta de los datos de la organización elegida. Ver 5.15.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre en los criterios de búsqueda. Ver 5.19.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.18.
- String **save**() guarda o actualiza el objeto `Organization` actual, volviendo a la misma página. Ver 5.16.
- String **search**() redirige al formulario con los criterios de búsqueda de organizaciones (`/pages/contacts/searchOrganization.jsp`).

- String **editarPrecios**() selecciona la organización, y si el usuario tiene permiso de modificarlo, lo redirige a la página de edición de precios. Ver 5.20.
- String **createEnsayoPrecios**() añade un nuevo particular precio de ensayo para esta organización y vuelve a la misma página. Ver 5.21
- String **deleteEnsayoPrecios**() elimina el particular precio de ensayo y vuelve a la misma página. Ver 5.22
- String **createPautaPrecios**() añade un nuevo particular precio de pauta para esta organización y vuelve a la misma página. Ver 5.23
- String **deletePautaPrecios**() elimina el particular precio de pauta y vuelve a la misma página. Ver 5.24

En la figura 5.14 se observa cómo se contruye una organización nueva.

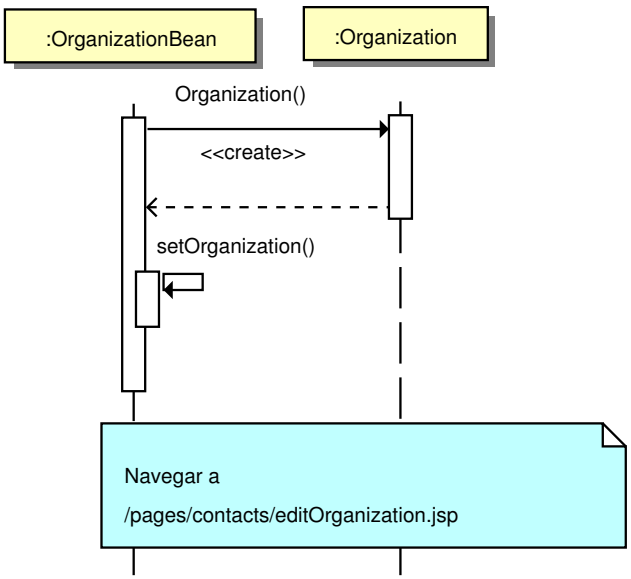

Figura 5.14: Diagrama de secuencia: Crear organización

Para editar la organización, hay que recuperar el objeto dado su id, y luego devolverlo a la vista. Véase la figura 5.15.

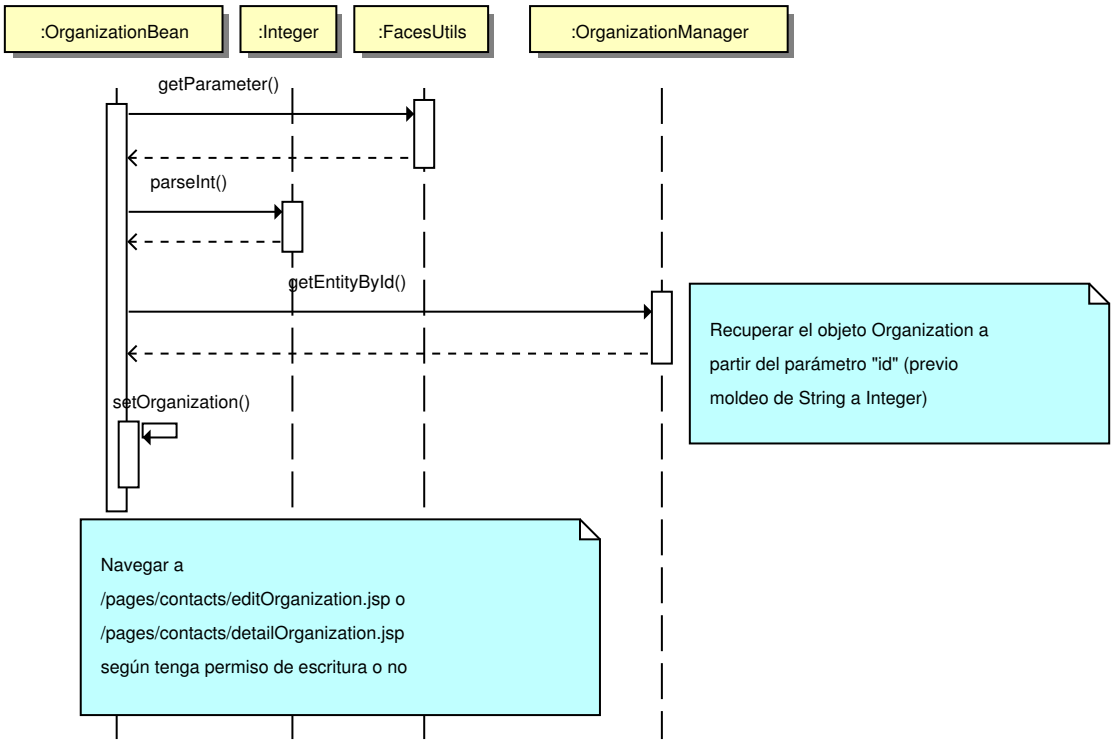

Figura 5.15: Diagrama de secuencia: Editar organización

En la figura 5.16 se muestra la secuencia para guardar una organización, si es nueva hay que insertarla, y si ya existe, actualizar la persistencia de su estado interno.

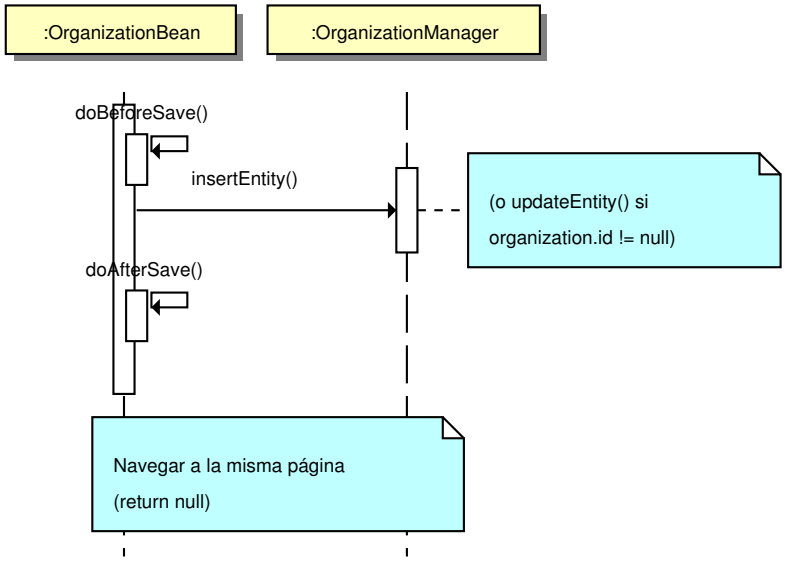

Figura 5.16: Diagrama de secuencia: Guardar organización

La secuencia necesaria para borrar una organización se puede ver a continuación, en la figura 5.17

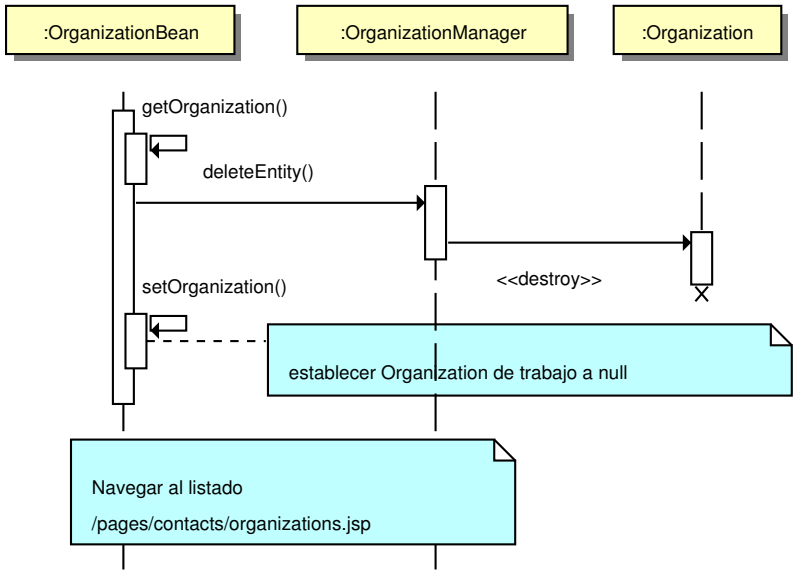

Figura 5.17: Diagrama de secuencia: Borrar organización

La figura 5.18 muestra la secuencia para eliminar los filtros de búsqueda.

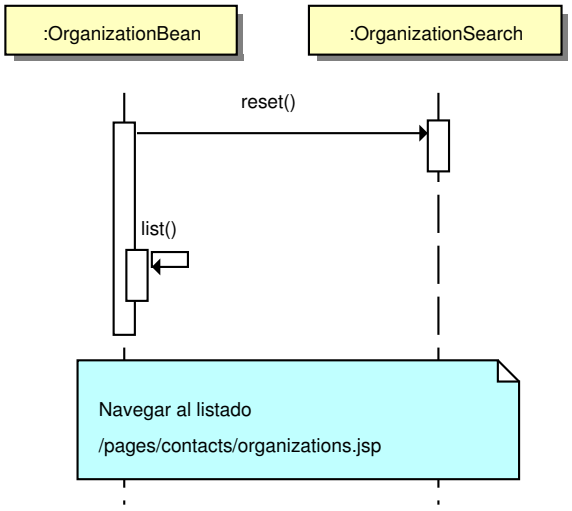

Figura 5.18: Diagrama de secuencia: Resetear búsqueda de organizaciones

La figura 5.19 muestra la secuencia para buscar por letra.

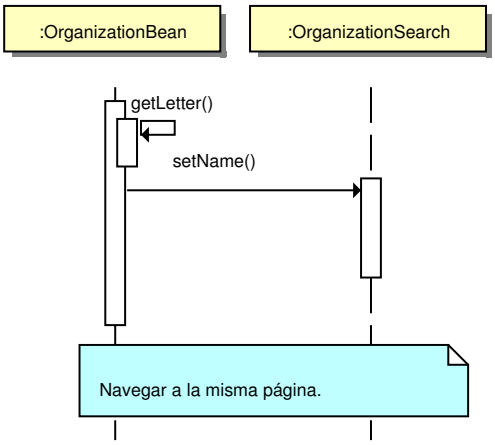

Figura 5.19: Diagrama de secuencia: Paginar organizaciones por letra

En la figura 5.20 diseñamos cómo mostrar la pantalla de edición de precios de la organización.

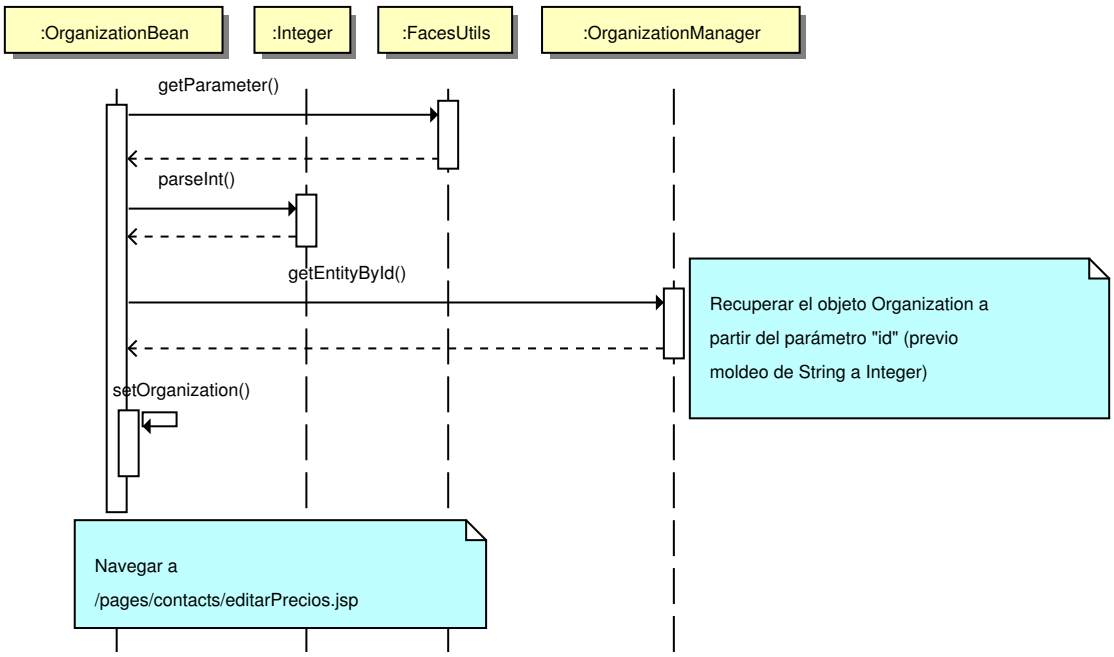

Figura 5.20: Diagrama de secuencia: Editar precios

La secuencia para añadir un precio particular de ensayo en blanco se puede ver en la figura 5.21

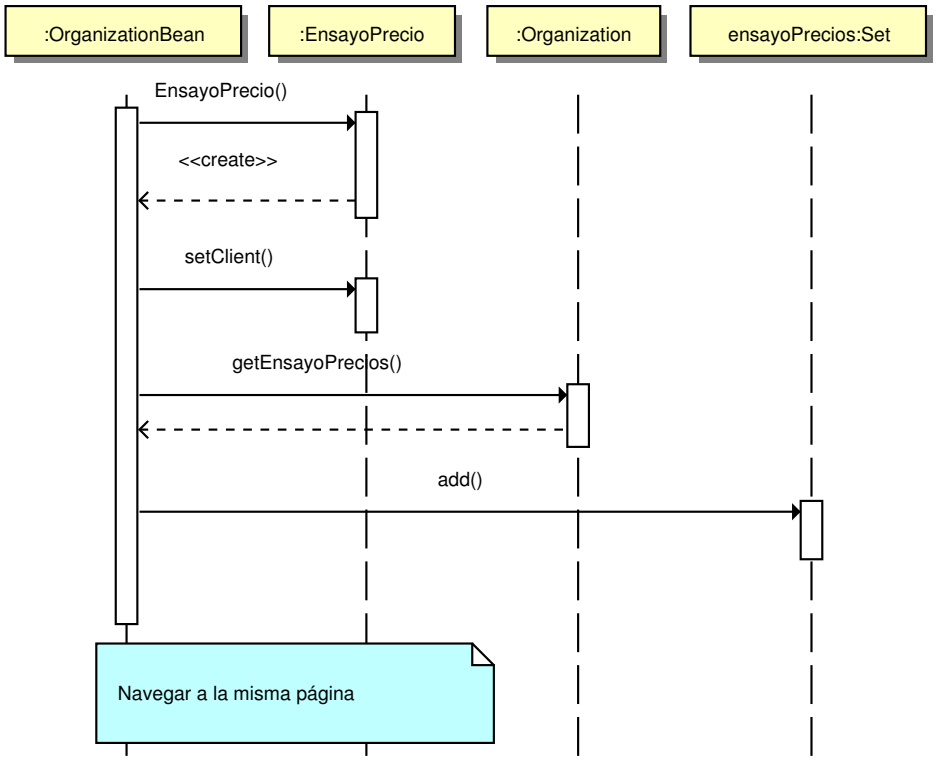

Figura 5.21: Diagrama de secuencia: Añadir particular precio de ensayo

La figura 5.22 muestra cómo eliminar el precio particular de ensayo determinado de la lista de precios particulares.

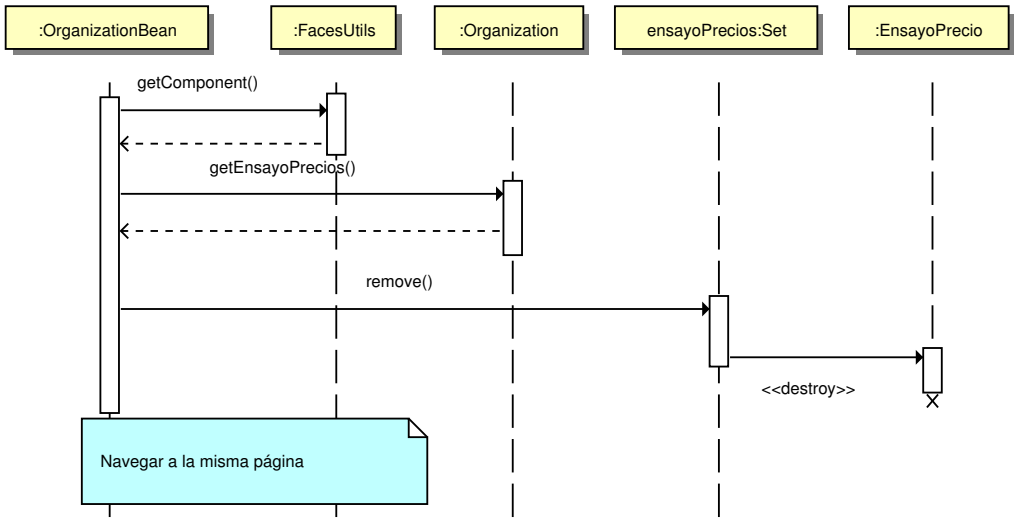

Figura 5.22: Diagrama de secuencia: Eliminar particular precio de ensayo

La secuencia para añadir un precio particular de pauta en blanco se muestra en la figura 5.23

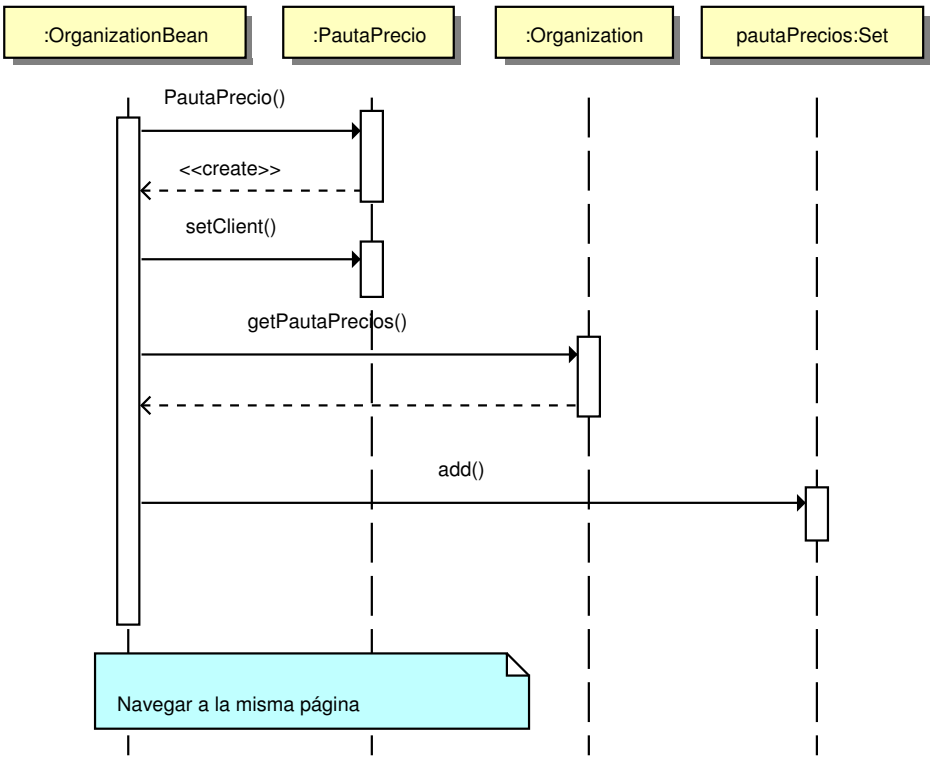

Figura 5.23: Diagrama de secuencia: Añadir particular precio de pauta

La figura 5.24 muestra cómo eliminar el precio particular de pauta determinado de la lista de precios particulares.

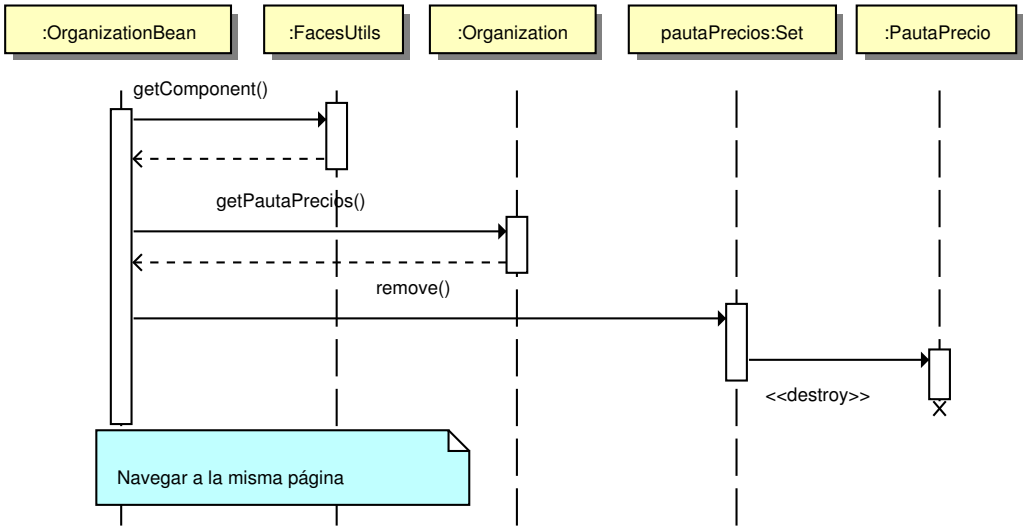

Figura 5.24: Diagrama de secuencia: Eliminar particular precio de pauta

5.3.3.2. ContactBean

Back-bean para las pantallas de gestión de contactos:

- `/pages/contacts/detailContact.jsp`
- `/pages/contacts/editContact.jsp`
- `/pages/contacts/searchContact.jsp`
- `/pages/contacts/contacts.jsp`

Propiedades y métodos:

- (RW) Contact **contact** contacto de trabajo activo.
- (R) Boolean **contactSelected** indica si se ha elegido algún contacto.
- (R) ContactSearch **search** objeto con los criterios de búsqueda para contactos.
- static ContactManager **manager** gestor de objetos de tipo `Contact`.
- String **create**() crea un nuevo contacto en blanco y redirige a la página de edición. Véase 5.25.
- String **delete**() deselecciona el contacto actual, lo elimina, y regresa al listado. Véase 5.28.
- String **detail**() va a la página de edición o de consulta de los datos del contacto elegido. Ver 5.26.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre en los criterios de búsqueda. Ver 5.30.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.29.
- String **save**() guarda o actualiza el objeto `Contact` actual, volviendo a la misma página. Ver 5.27.
- String **search**() redirige al formulario con los criterios de búsqueda de contactos (`/pages/contacts/searchContact.jsp`).

La secuencia para crear un contacto, es simplemente llamar a su constructor y devolverlo a la vista, como se puede ver en la figura 5.25

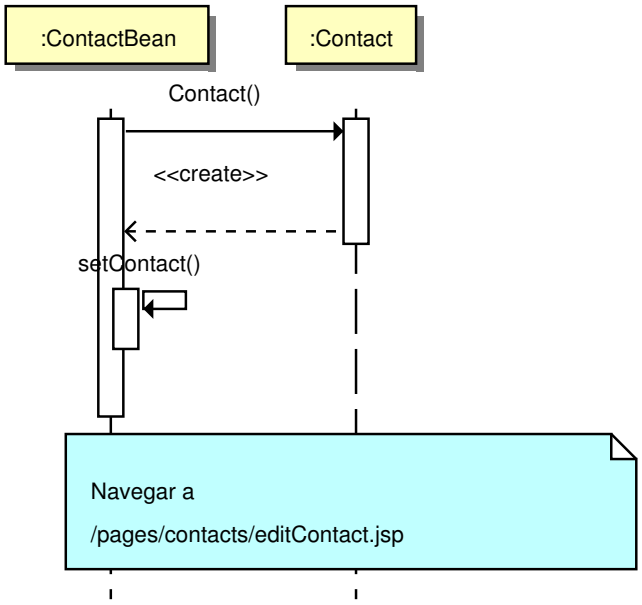

Figura 5.25: Diagrama de secuencia: Crear contacto

Para editar un contacto, recuperarlo a partir de su id y devolverlo a la vista de edición (véase figura 5.26).

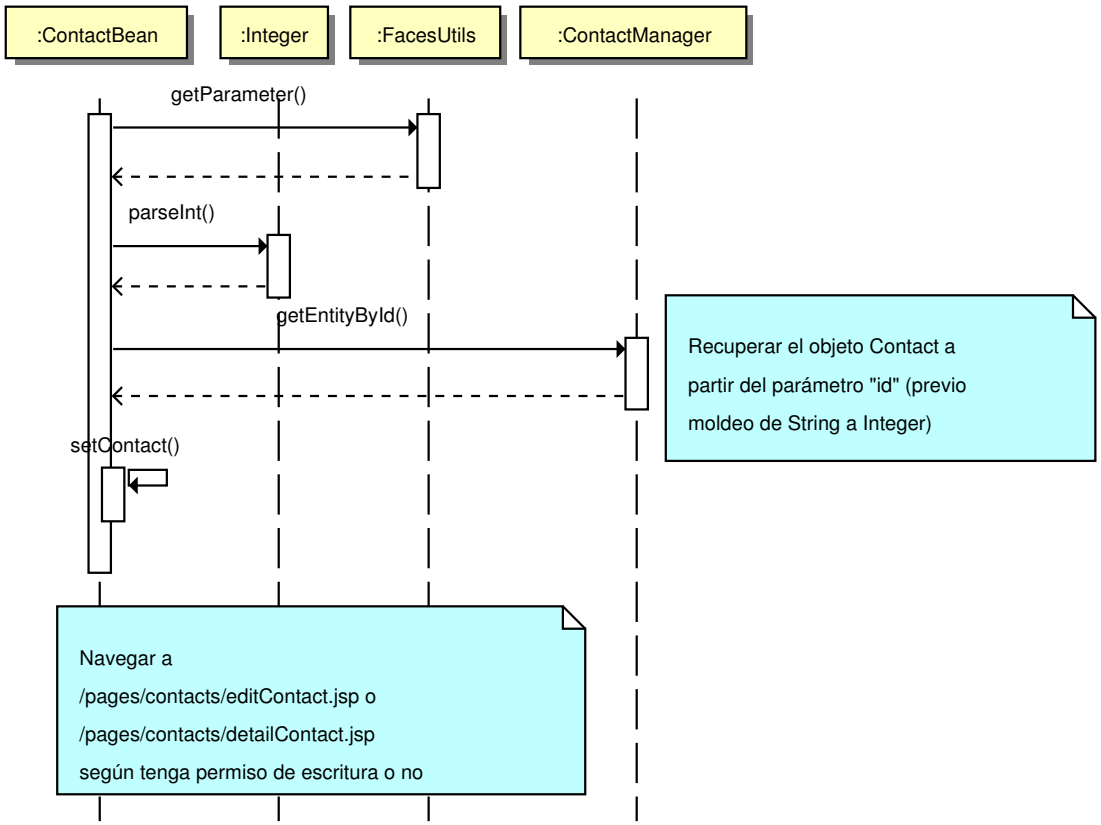

Figura 5.26: Diagrama de secuencia: Editar contacto

La secuencia para guardar o actualizar contactos se puede ver en la figura 5.27.

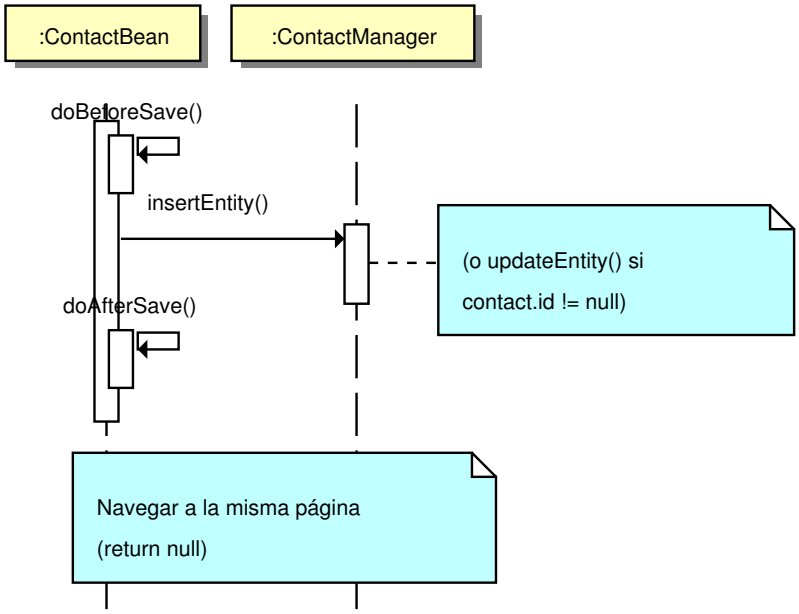

Figura 5.27: Diagrama de secuencia: Guardar contacto

Para borrar un contacto, sólo hay que destruirlo y establecer null como contacto actual en la vista (véase figura 5.28).

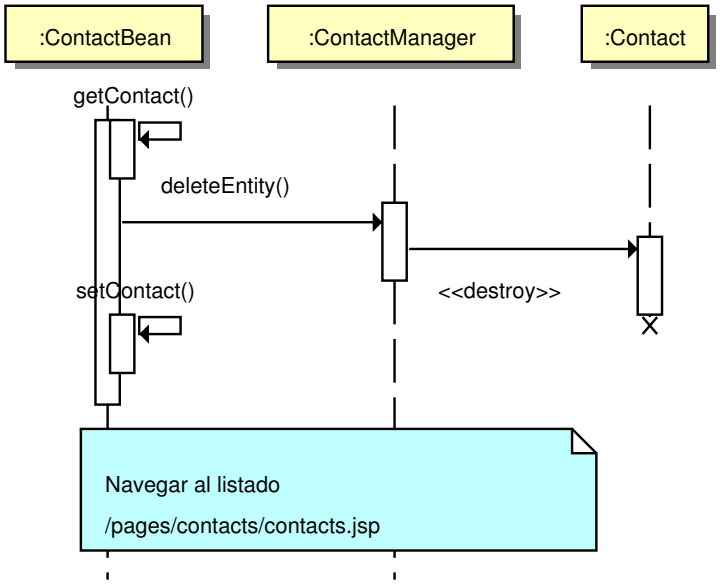

Figura 5.28: Diagrama de secuencia: Borrar contacto

En la figura 5.29 se detalla cómo restablecer los criterios de búsqueda.

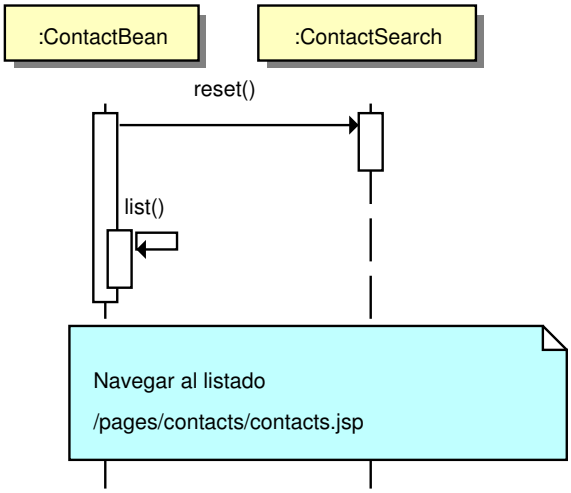

Figura 5.29: Diagrama de secuencia: Resetear búsqueda de contactos

La secuencia para paginar por letra, es estableciéndola como criterio de búsqueda, como se puede ver en la figura 5.30

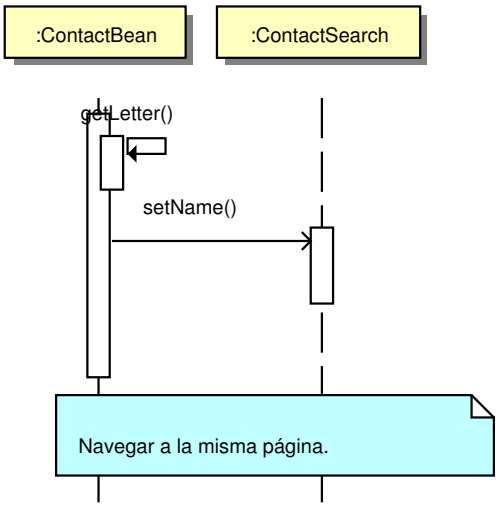

Figura 5.30: Diagrama de secuencia: Paginar contactos por letra

5.3.3.3. OfferBean

Back-bean para las pantallas de gestión de presupuestos:

- `/pages/contacts/detailOffer.jsp`
- `/pages/contacts/editOffer.jsp`
- `/pages/contacts/searchOffer.jsp`
- `/pages/contacts/offers.jsp`

Propiedades y métodos:

- (RW) Offer **offer** presupuesto de trabajo activo.
- (R) Boolean **offerSelected** indica si se ha elegido algún presupuesto o no.
- (R) OfferSearch **search** objeto con los criterios de búsqueda para presupuestos.
- static OfferManager **manager** gestor de objetos de tipo `Offer`.
- String **create**() crea un nuevo presupuesto en blanco y redirige a la página de edición. Véase 5.31.
- String **delete**() deselecciona el presupuesto actual, lo elimina, y regresa al listado. Véase 5.34.
- String **detail**() va a la página de edición o de consulta de los datos del presupuesto elegido. Ver 5.32.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del título en los criterios de búsqueda. Ver 5.36.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.35.
- String **save**() guarda o actualiza el objeto `Offer` actual, volviendo a la misma página. Ver 5.33.
- String **search**() redirige al formulario con los criterios de búsqueda de presupuestos (`/pages/contacts/searchOffer.jsp`).

Para crear un presupuesto nuevo, hay que invocar al constructor vacío, y después completar los campos obligatorios con los valores por defecto, como se ve en su diagrama de secuencia (figura 5.31)

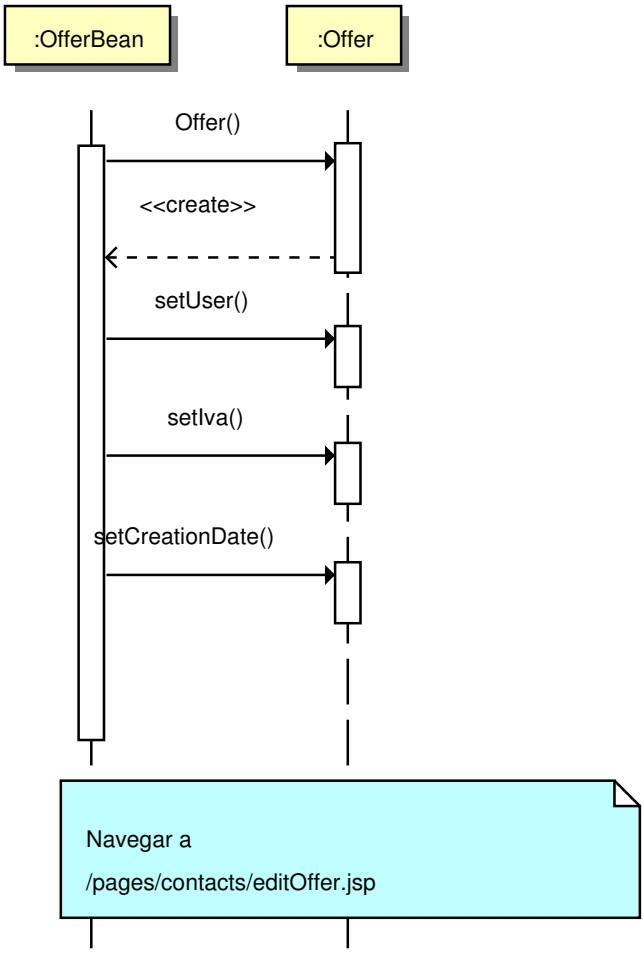

Figura 5.31: Diagrama de secuencia: Crear presupuesto

La secuencia para editar un presupuesto es recuperarlo y pasarlo a la vista (véase figura 5.32).

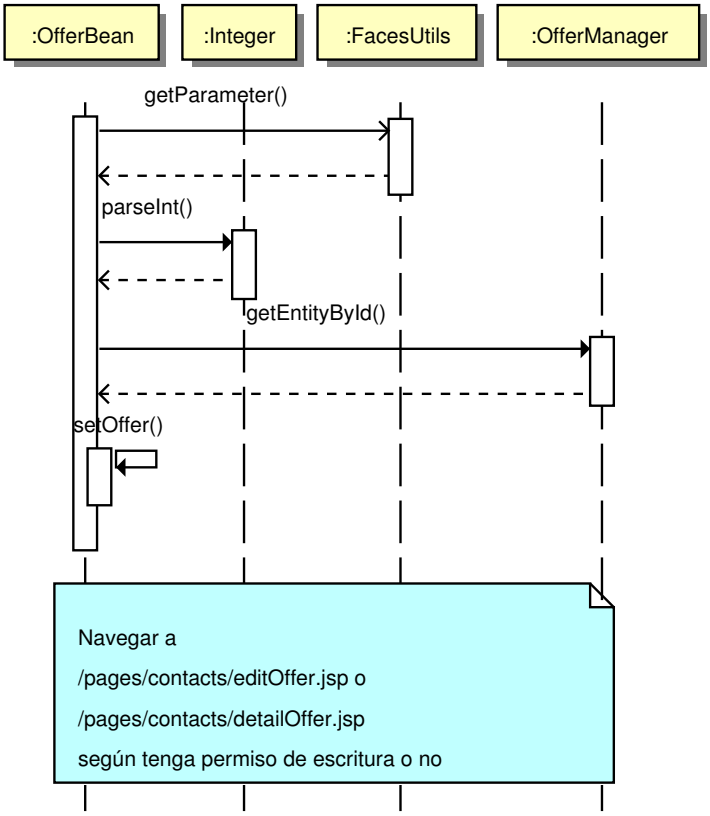

Figura 5.32: Diagrama de secuencia: Editar presupuesto

En la figura 5.33 se puede ver la forma de insertar o actualizar el objeto.

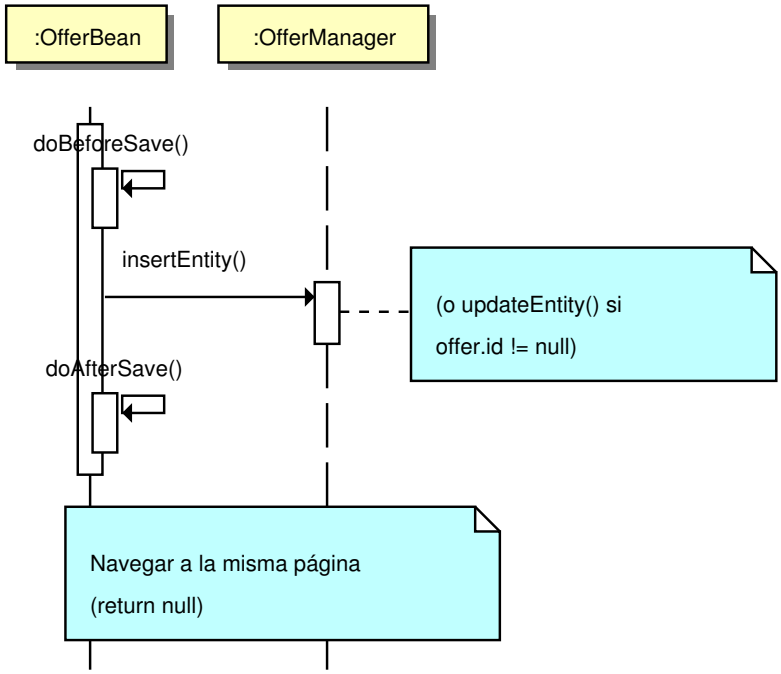

Figura 5.33: Diagrama de secuencia: Guardar presupuesto

La figura 5.34 muestra cómo destruir el presupuesto, eliminándolo de la base de datos y retirándolo de la vista.

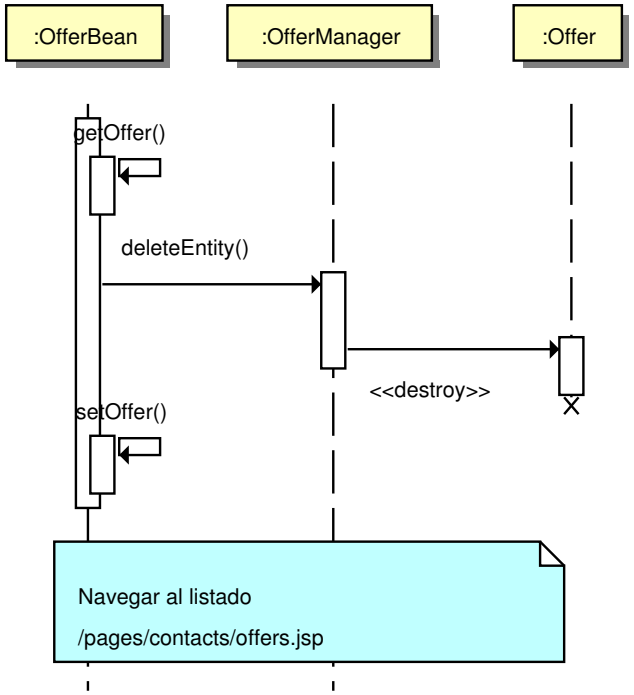

Figura 5.34: Diagrama de secuencia: Borrar presupuesto

La figura 5.35 especifica la secuencia para restablecer los criterios de búsqueda.

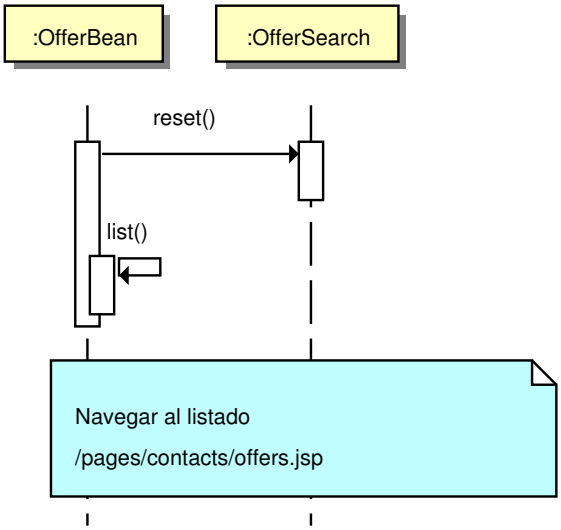

Figura 5.35: Diagrama de secuencia: Resetear búsqueda de presupuestos

Para paginar o buscar por letra, seguir la secuencia de la figura 5.36.

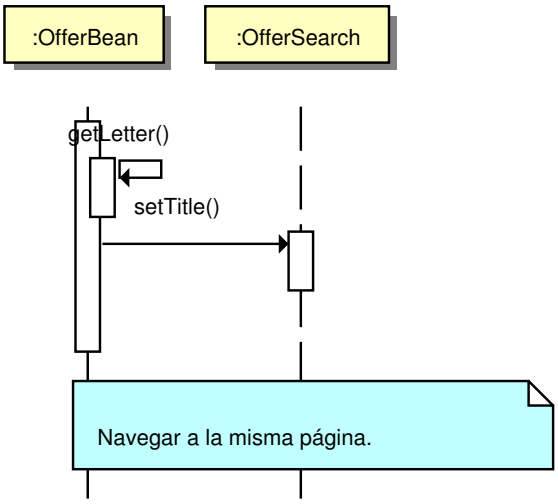

Figura 5.36: Diagrama de secuencia: Paginar presupuestos por letra

5.3.3.4. PautaBean

Back-bean para las pantallas de gestión de pautas:

- `/pages/contacts/detailPauta.jsp`
- `/pages/contacts/editPauta.jsp`
- `/pages/contacts/searchPauta.jsp`
- `/pages/contacts/pautas.jsp`

Propiedades y métodos:

- (RW) Pauta **pauta** pauta de trabajo activa.
- (R) Boolean **pautaSelected** indica si se ha elegido alguna pauta o no.
- (R) PautaSearch **search** objeto con los criterios de búsqueda para pautas.
- static PautaManager **manager** gestor de objetos de tipo `Pauta`.
- String **create**() crea una nueva pauta en blanco y redirige a la página de edición. Véase 5.37.
- String **delete**() deselecciona la pauta actual, la elimina, y regresa al listado. Véase 5.40.
- String **detail**() va a la página de edición o de consulta de los datos de la pauta elegida. Ver 5.38.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre de la pauta en los criterios de búsqueda. Ver 5.42.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.41.
- String **save**() guarda o actualiza el objeto `Pauta` actual, volviendo a la misma página. Ver 5.39.
- String **search**() redirige al formulario con los criterios de búsqueda de pautas (`/pages/contacts/searchPauta.jsp`).
- String **duplicar**() crea una pauta nueva como un réplica de éste, pasando al formulario de edición para la pauta nueva. Ver 5.43.

La secuencia para crear una pauta, es invocar al constructor por defecto, y devolver el objeto a la vista (véase figura 5.37).

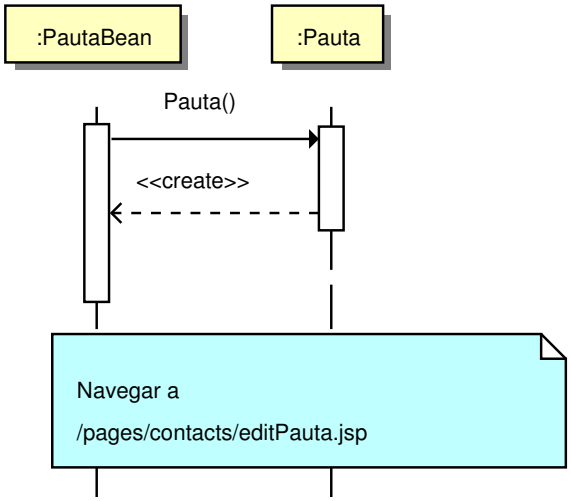

Figura 5.37: Diagrama de secuencia: Crear pauta

La secuencia para editar una pauta existente (véase figura 5.38), es recuperarla a partir de su identificador, y devolverlo a la vista del formulario de edición.

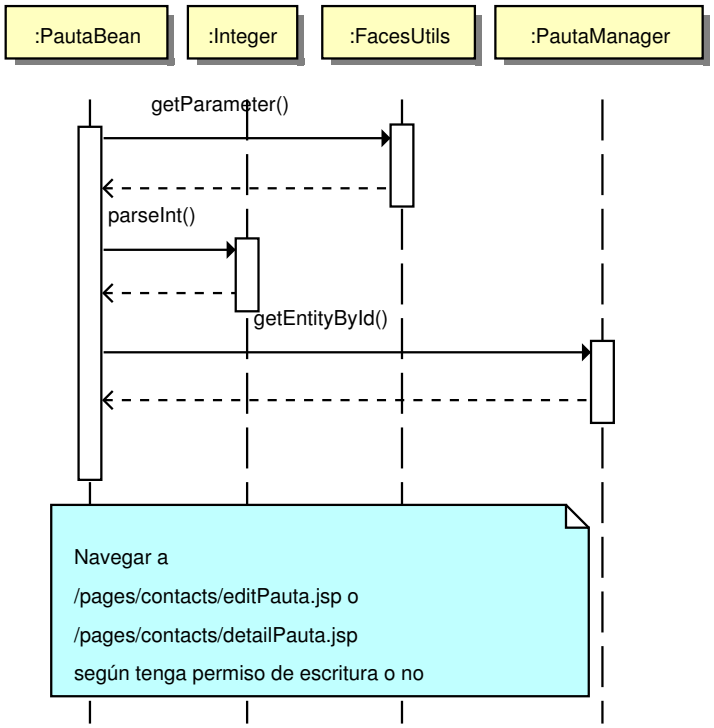

Figura 5.38: Diagrama de secuencia: Editar pauta

Para guardar una pauta, hay que insertarla o actualizarla, según exista o no, y devolver la misma vista (véase figura 5.39).

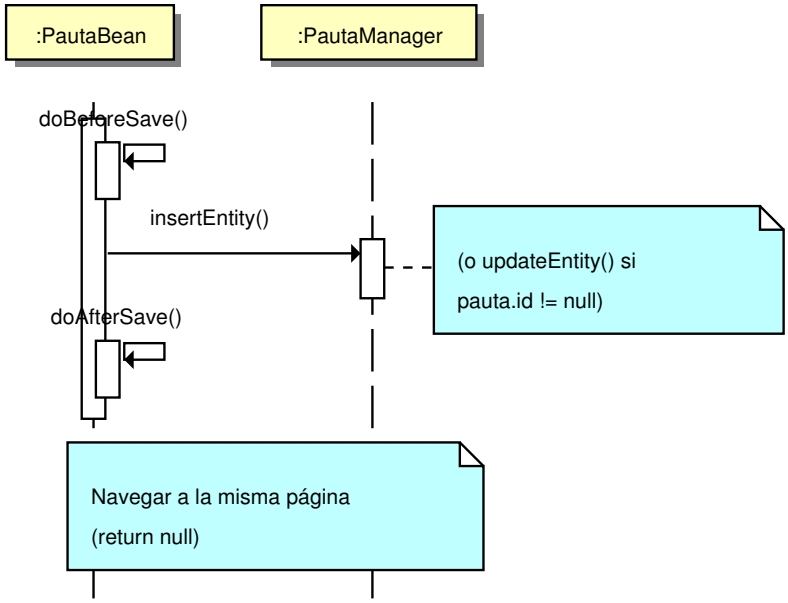

Figura 5.39: Diagrama de secuencia: Guardar pauta

La secuencia para borrar la pauta de trabajo se muestra en la figura 5.40: simplemente borrándolo de la base de datos y destruyendo el objeto.

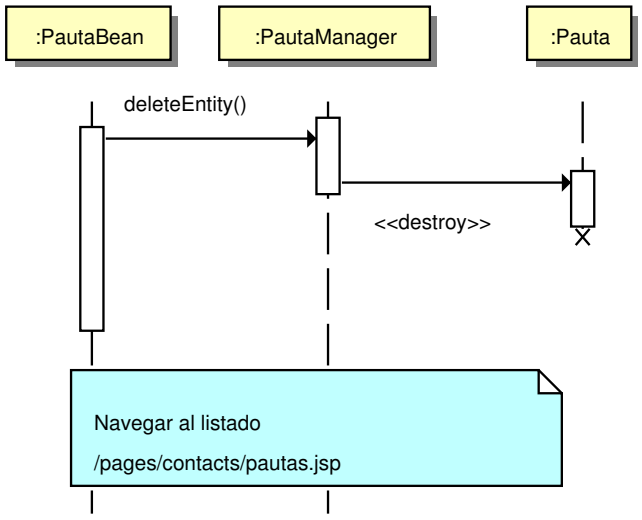

Figura 5.40: Diagrama de secuencia: Borrar pauta

La figura 5.41 muestra la secuencia para resetear los criterios de búsqueda, regresando a la vista del listado.

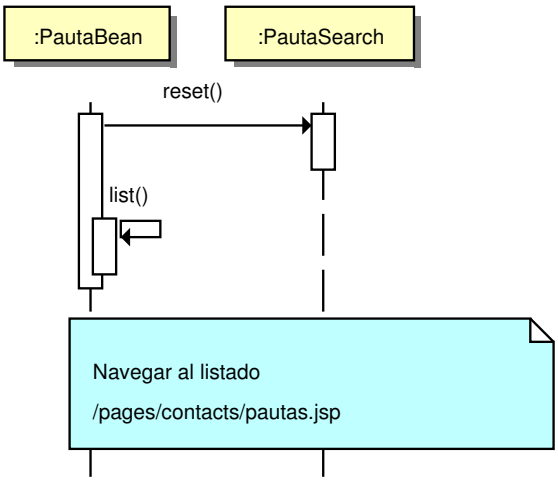

Figura 5.41: Diagrama de secuencia: Resetear búsqueda de pautas

La figura 5.42 especifica la forma de paginar el listado por letras: simplemente añadiendo un criterio de búsqueda y devolviendo a la vista del listado.

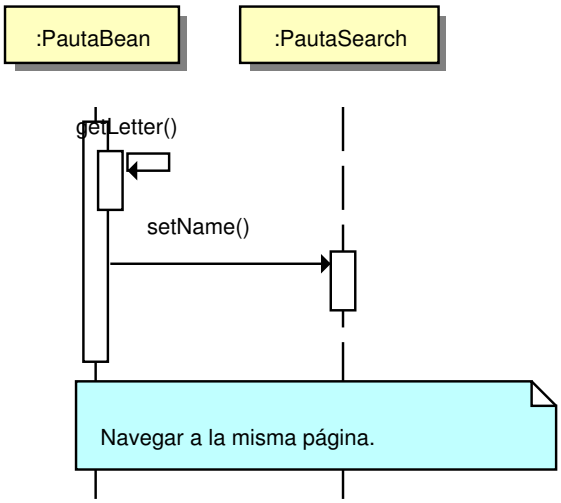

Figura 5.42: Diagrama de secuencia: Paginar pautas por letra

En la secuencia para duplicar una pauta (véase figura 5.43), primero hay que recuperar la pauta original, después crear una nueva en blanco, y a continuación ir copiando todos sus atributos,

(excepto el nombre, que se le pondrá un sufijo). Insertar el nuevo en la base de datos y devolver la vista a la página de edición de la pauta nueva.

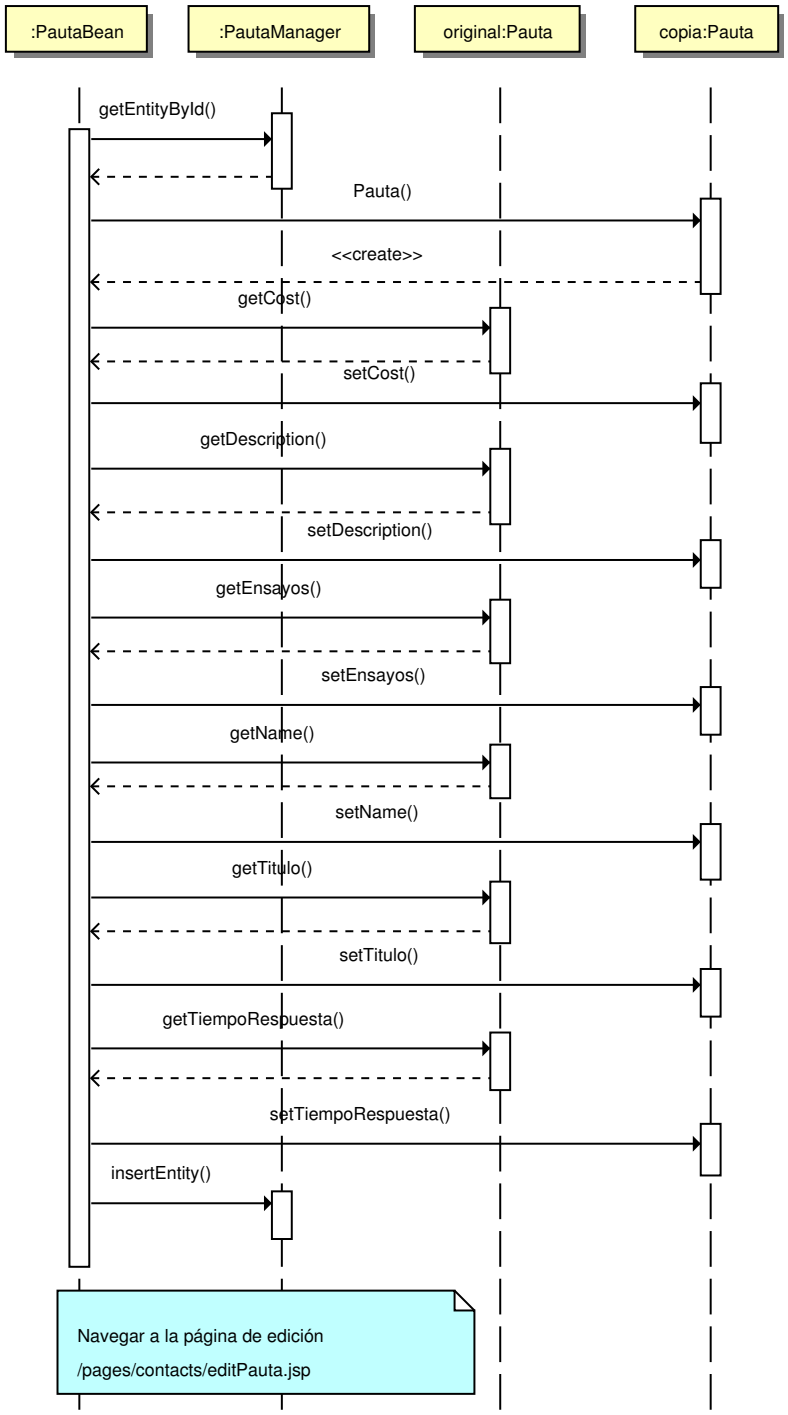

Figura 5.43: Diagrama de secuencia: Duplicar pauta

5.3.3.5. EnsayoBean

Back-bean para las pantallas de gestión de ensayos:

- `/pages/contacts/detailEnsayo.jsp`
- `/pages/contacts/editEnsayo.jsp`
- `/pages/contacts/searchEnsayo.jsp`
- `/pages/contacts/ensayos.jsp`

Propiedades y métodos:

- (RW) Ensayo **ensayo** ensayo de trabajo activo.
- (R) Boolean **ensayoSelected** indica si se ha elegido algún ensayo o no.
- (R) EnsayoSearch **search** objeto con los criterios de búsqueda para ensayos.
- static EnsayoManager **manager** gestor de objetos de tipo `Ensayo`.
- String **create**() crea un nuevo ensayo en blanco y redirige a la página de edición. Véase 5.44.
- String **delete**() deselecciona el ensayo actual, lo elimina, y regresa al listado. Véase 5.47.
- String **detail**() va a la página de edición o de consulta de los datos del ensayo elegido. Ver 5.45.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre del ensayo en los criterios de búsqueda. Ver 5.49.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.48.
- String **save**() guarda o actualiza el objeto `Ensayo` actual, volviendo a la misma página. Ver 5.46.
- String **search**() redirige al formulario con los criterios de búsqueda de ensayos (`/pages/contacts/searchEnsayo.jsp`).

Para crear un ensayo (véase figura 5.44), hay que construir uno nuevo, establecerlo como ensayo de trabajo y devolver a la vista de edición de este nuevo ensayo.

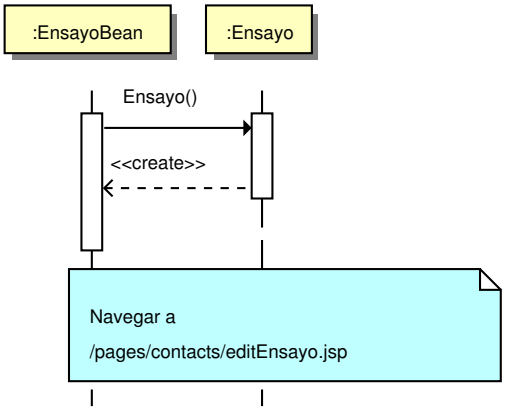

Figura 5.44: Diagrama de secuencia: Crear ensayo

En la figura 5.45 se puede ver la secuencia para editar un ensayo: recuperar el objeto a partir de su identificador, y devolver la vista con el formulario de edición.

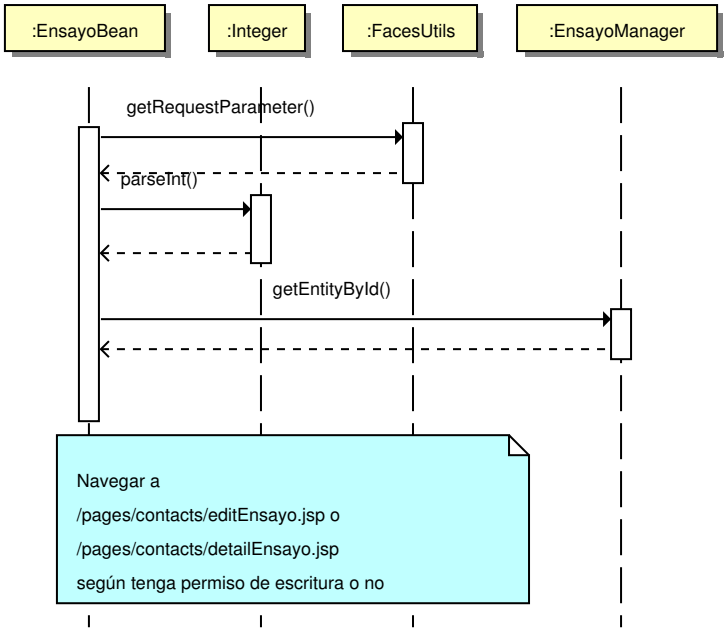

Figura 5.45: Diagrama de secuencia: Editar ensayo

La figura 5.46 define la secuencia para insertar un ensayo nuevo o actualizar uno existente.

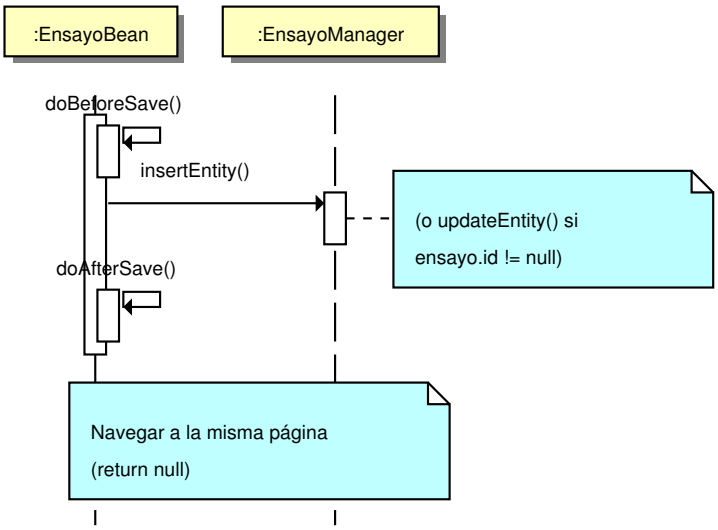

Figura 5.46: Diagrama de secuencia: Guardar ensayo

La figura 5.47 muestra la secuencia para borrar un ensayo: eliminándolo de la base de datos y destruyendo el objeto.

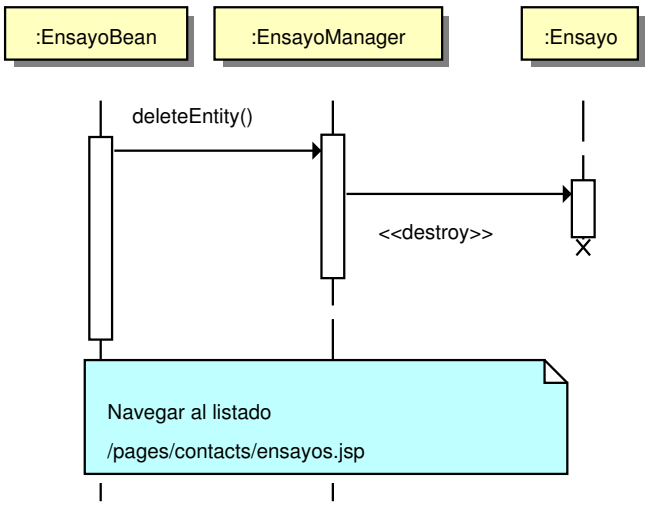

Figura 5.47: Diagrama de secuencia: Borrar ensayo

La figura 5.48 muestra la secuencia para restablecer los criterios de búsqueda y volver al listado.

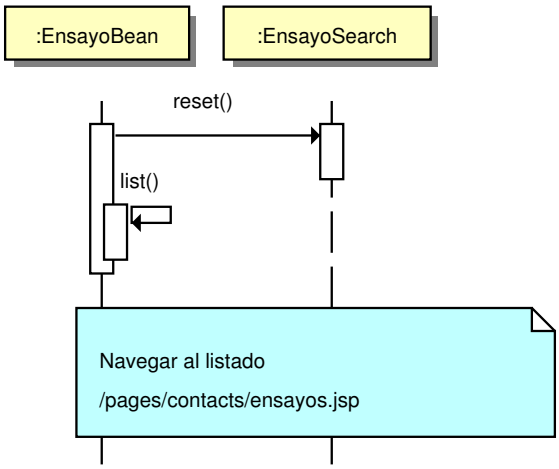

Figura 5.48: Diagrama de secuencia: Resetear búsqueda de ensayos

En la figura 5.49 se observa que para paginar el listado por letra iniciar, sólo hay que añadir el criterio de búsqueda, y repetir el listado.

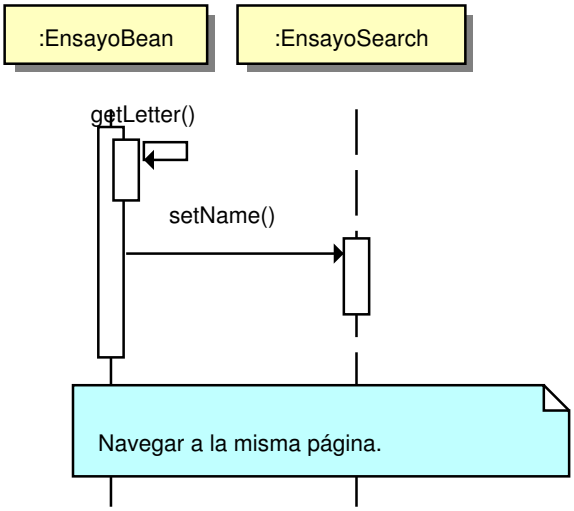

Figura 5.49: Diagrama de secuencia: Paginar ensayos por letra

5.3.3.6. ActualizarPreciosBean

Back-bean para el actualizador de precios en masa:

- `/pages/contacts/actualizarPrecios.jsp`

Propiedades y métodos:

- (RW) List<String>**actualizar** conjunto de opciones a actualizar: sólo ensayos, sólo pautas o ambos.
- (RW) List<Organization>**clientes** conjunto de clientes para los que actualizar precios.
- (RW) BigDecimal **factor** factor de automultiplicación de los precios particulares.
- String **run**() efectúa la actualización siguiendo las opciones seleccionadas. Véase 5.50.

La figura 5.50 describe la secuencia para actualizar precios en lote: recuperar la lista de ensayos y automultiplicar su coste por el factor. A continuación, recuperar la lista de pautas, y

automultiplicar su coste por el factor.

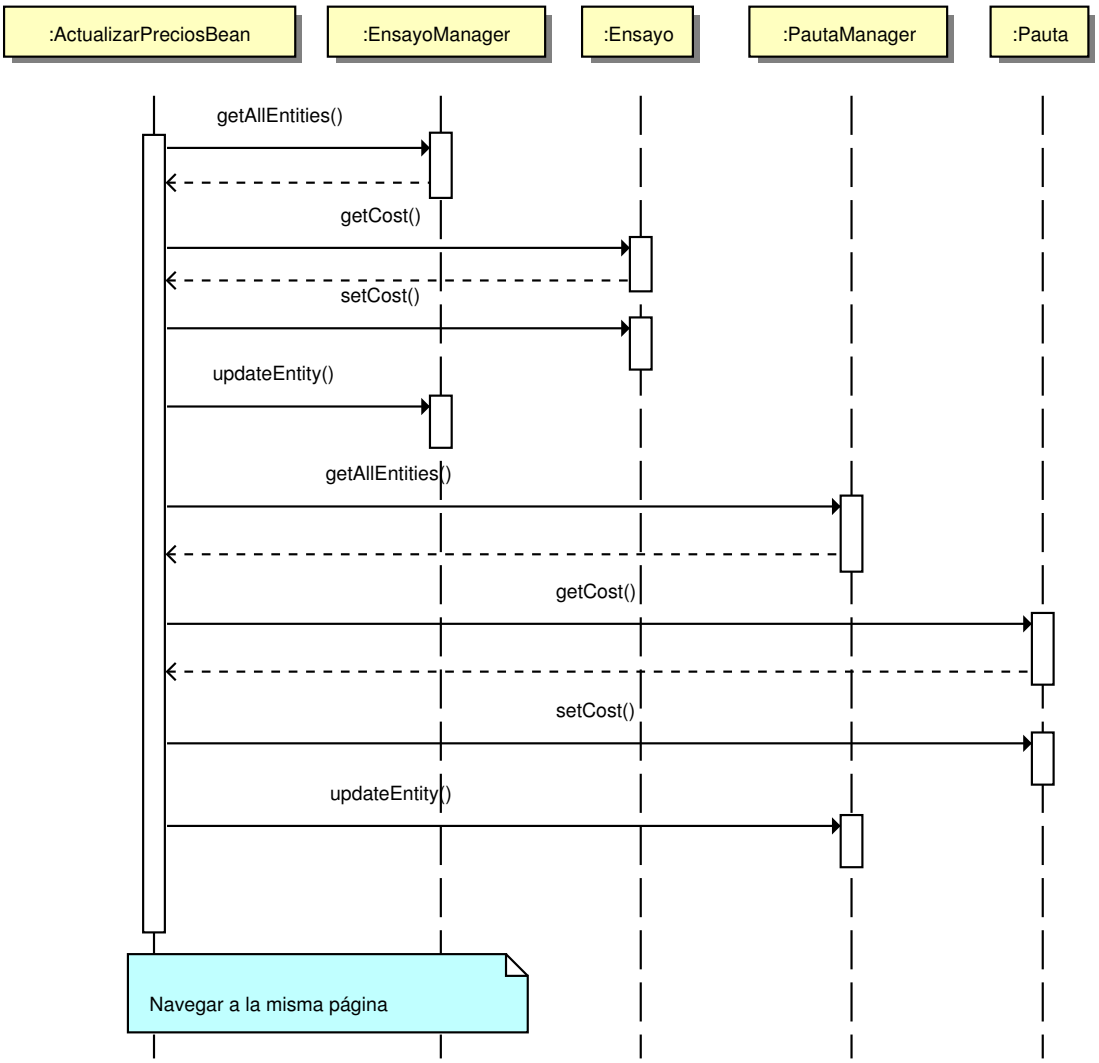

Figura 5.50: Diagrama de secuencia: Actualizar precios

5.3.3.7. AlbaranBean

Back-bean para las pantallas de gestión de albaranes:

- `/pages/contacts/detailAlbaran.jsp`
- `/pages/contacts/editAlbaran.jsp`
- `/pages/contacts/searchAlbaran.jsp`
- `/pages/contacts/albarans.jsp`

Propiedades y métodos:

- (RW) Albaran **albaran** albarán de trabajo activo.
- (R) Boolean **albaranSelected** indica si existe albarán elegido.
- (R) AlbaranSearch **search** objeto con los criterios de búsqueda para albaranes.
- static AlbaranManager **manager** gestor de objetos de tipo `Albaran`.
- String **create**() crea un nuevo albarán en blanco y redirige a la página de edición. Véase 5.51.
- String **delete**() borra el albarán actual y vuelve a la página del listado. Véase 5.54.
- String **detail**() va a la página de edición o de consulta de los datos del albarán elegido. Ver 5.52.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre en los criterios de búsqueda. Ver 5.56.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.55.
- String **save**() guarda o actualiza el objeto `Albaran` actual, volviendo a la misma página. Ver 5.53.
- String **search**() redirige al formulario con los criterios de búsqueda de albaranes (`/pages/contacts/searchAlbaran.jsp`).

La figura 5.51 muestra la secuencia para crear un albarán nuevo: construirlo y devolverlo a la vista de edición, para comenzar a completar sus campos.

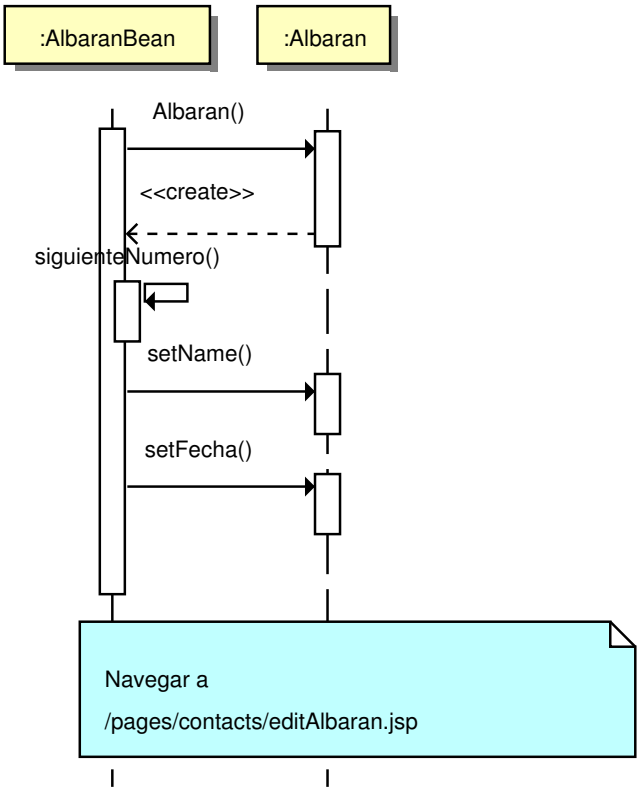

Figura 5.51: Diagrama de secuencia: Crear albarán

La figura 5.52 muestra la secuencia para editar un albarán existente: recuperarlo a partir de su identificador, y devolver la vista de edición.

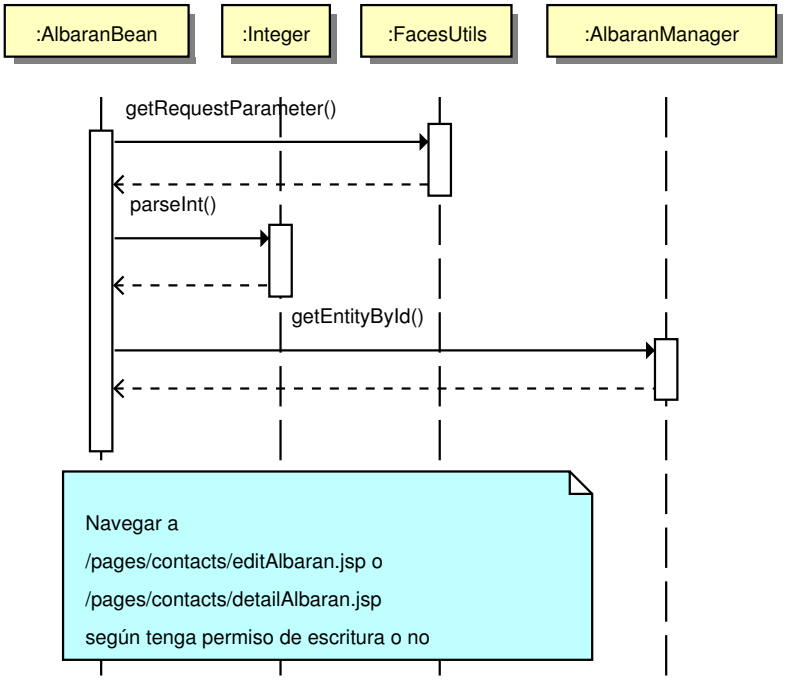

Figura 5.52: Diagrama de secuencia: Editar albarán

La figura 5.53 describe la secuencia para insertar o actualizar un albarán en la base de datos.

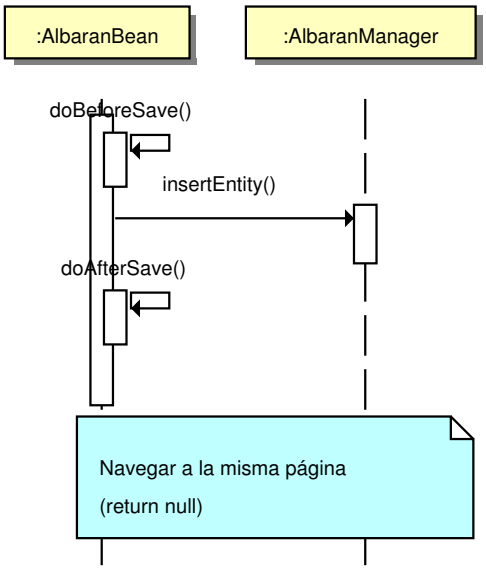

Figura 5.53: Diagrama de secuencia: Guardar albarán

La figura 5.54 describe la secuencia para borrar un albarán de la base de datos, destruyendo el objeto, y regresando la vista al listado de albaranes.

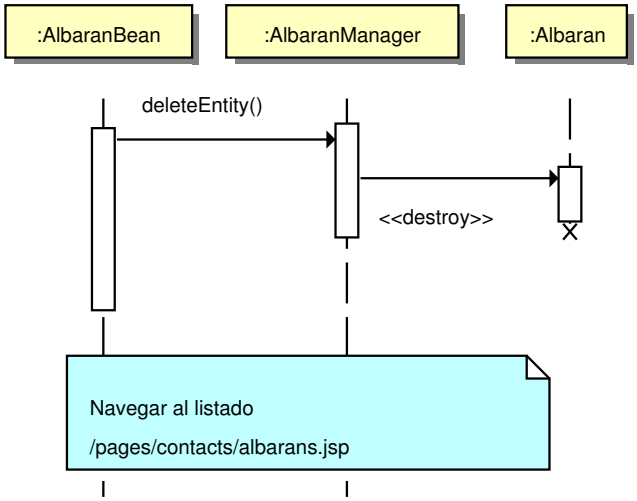

Figura 5.54: Diagrama de secuencia: Borrar albarán

En la figura 5.55 se puede observar la secuencia necesaria para limpiar los criterios de búsqueda y devolver la vista del listado con todos los albaranes.

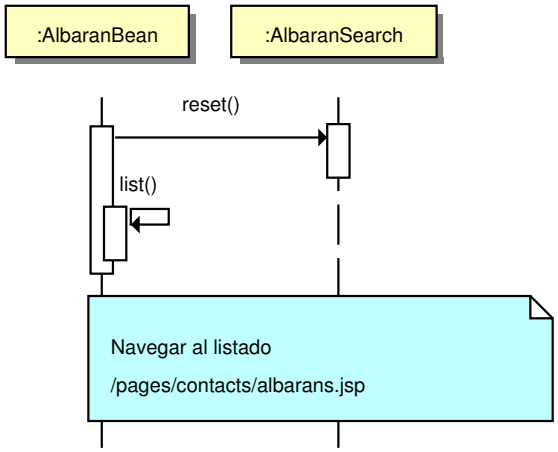

Figura 5.55: Diagrama de secuencia: Resetear búsqueda de albaranes

La figura 5.56 describe la secuencia para filtrar por letra inicial del nombre del albarán.

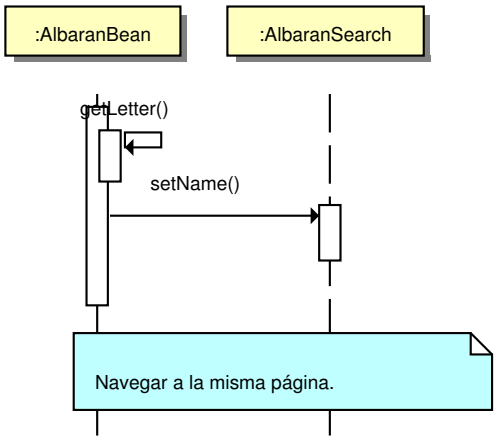

Figura 5.56: Diagrama de secuencia: Paginar albaranes por letra

5.3.3.8. NotaSalidaBean

Back-bean para las pantallas de gestión de notas de salida:

- `/pages/contacts/detailNotaSalida.jsp`
- `/pages/contacts/editNotaSalida.jsp`
- `/pages/contacts/searchNotaSalida.jsp`
- `/pages/contacts/notaSalidas.jsp`

Propiedades y métodos:

- (RW) NotaSalida **notaSalida** nota de salida de trabajo activa.
- (R) Boolean **notaSalidaSelected** indica si existe nota de salida elegida.
- (R) NotaSalidaSearch **search** objeto con los criterios de búsqueda para notas de salida.
- static NotaSalidaManager **manager** gestor de objetos de tipo `NotaSalida`.
- String **create**() crea una nueva nota de salida en blanco y redirige a la página de edición. Véase 5.57.
- String **delete**() borra la nota de salida actual y vuelve a la página de listado. Véase 5.60.
- String **detail**() va a la página de edición o de consulta de los datos de la nota de salida elegida. Ver 5.58.
- (RW) Character **letter** letra inicial para el paginador alfabético.
- void **letterClicked**() establece la letra elegida como letra inicial del nombre en los criterios de búsqueda. Ver 5.62.
- String **list**() devuelve el control a la página del listado.
- String **reset**() restablece todos los criterios de búsqueda y devuelve la navegación a la página del listado. Ver 5.61.
- String **save**() guarda o actualiza el objeto `NotaSalida` actual, volviendo a la misma página. Ver 5.59.
- String **search**() redirige al formulario con los criterios de búsqueda de notas de salida (`/pages/contacts/searchNotaSalida.jsp`).

La figura 5.57 muestra la secuencia para crear una nueva nota de salida en blanco y editarla a continuación.

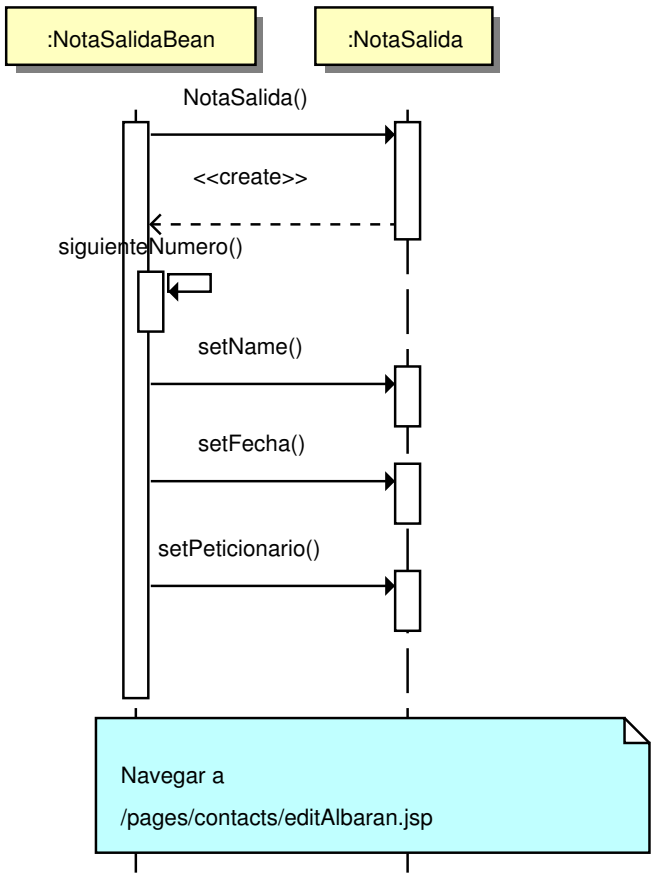

Figura 5.57: Diagrama de secuencia: Crear nota de salida

La secuencia para editar una nota de salida, recuperándola y pasándola a la vista del formulario de edición, se puede observar en la figura 5.58.

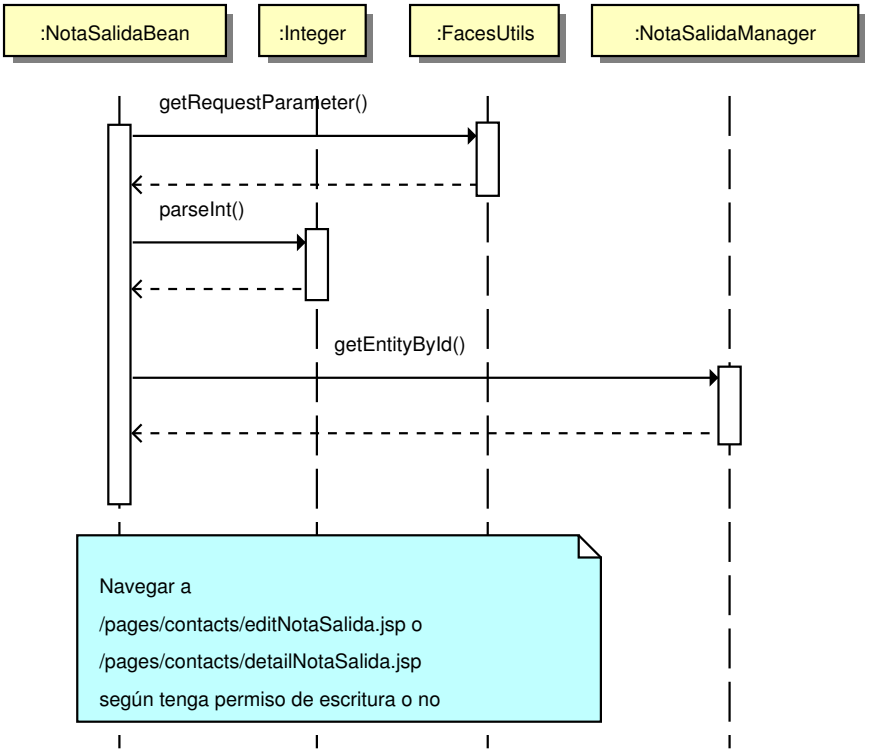

Figura 5.58: Diagrama de secuencia: Editar nota de salida

La secuencia para guardar insertando o actualizando notas de salida se encuentra en la figura 5.59.

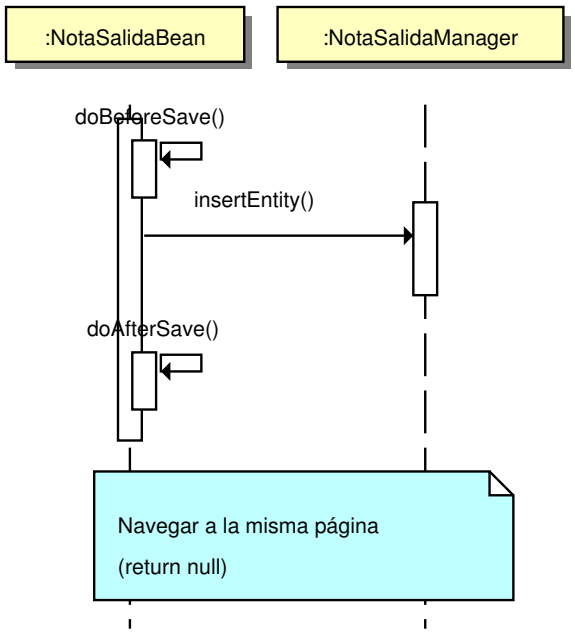

Figura 5.59: Diagrama de secuencia: Guardar nota de salida

Para borrar una nota de salida, hay que eliminarla de la base de datos y destruir el objeto, regresando la vista al listado de notas de salida (véase figura 5.60).

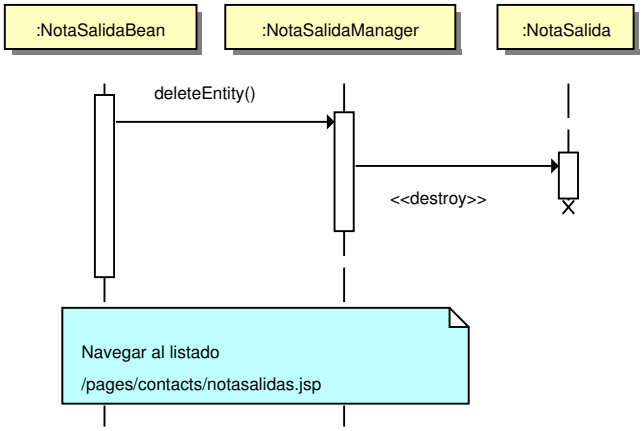

Figura 5.60: Diagrama de secuencia: Borrar nota de salida

Para resetar notas de salida, hay que vaciar los criterios de búsqueda y devolver al listado, como se describe en la figura 5.61.

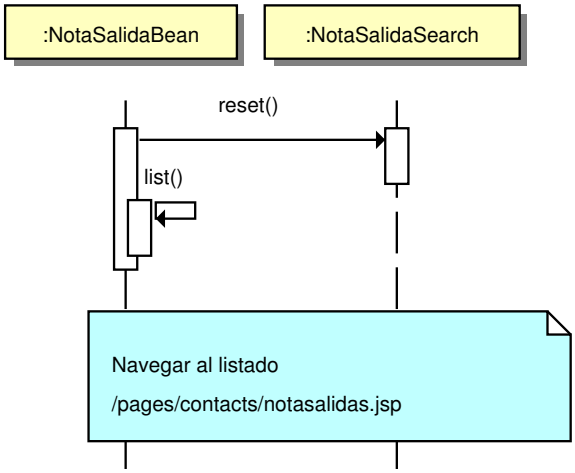

Figura 5.61: Diagrama de secuencia: Resetear búsqueda de notas de salida

Para paginar notas de salida por su letra inicial, hay que añadir el criterio de búsqueda y repetir el listado, como se observa en la figura 5.62.

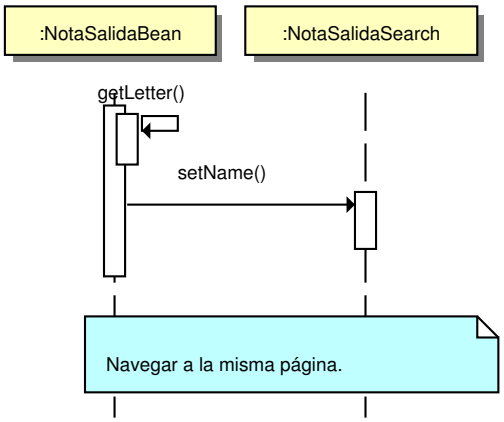

Figura 5.62: Diagrama de secuencia: Paginar notas de salida por letra

5.3.4. `com.autentia.intra.businessobject`

Este paquete aloja todas las clases que se deducen de los diagramas de clases que se presentan en la sección 5.2, que son clases del dominio del problema, más clases para representar las relaciones entre éstas, y algunos de los tipos enumerados:

- Albaran
- Bill
- BillBreakDown
- BillPaymentMode
- BillState
- BillType
- Contact
- ContractType
- Ensayo
- EnsayoPrecio
- EnsayoProcedimiento
- Historial
- HistorialType
- Interaction
- InteractionInterest
- InteractionType
- NotaSalida
- Offer
- OfferCost
- OfferEnsayo
- OfferPaura
- OfferPotential
- OfferRejectReason
- OfferRole
- OfferState
- Organization
- OrganizationCertificado
- OrganizationEstado
- OrganizationISOCategory
- OrganizationType
- ParametroString
- Pauta
- PautaEnsayo
- PautaEnsayoDimension
- PautaIdentificacion
- PautaPrecio
- Project
- ProjectCost
- ProjectEnsayo
- ProjectEnsayoDimension
- ProjectEnsayoIdentificacion
- ProjectEnsayoRequerimiento
- ProjectEstado
- ProjectIdentificacion
- ProjectRole
- Province
- Role
- User
- WorkingAgreement

5.3.5. `com.autentia.intra.businessobject.config`

Este paquete contiene la definición, un fichero XML por cada clase, de los modelos para el motor de persistencia Hibernate.

5.3.6. `com.autentia.intra.converter`

Las clases de este paquete implementan la interfaz `javax.faces.convert.Converter` cuyo contrato son los siguientes dos métodos. Éstos se emplean en la fase de renderizado de la vista, para convertir un objeto cualquiera a su representación como cadena, transformándolo como se necesite en cada momento:

- *Object getAsObject(FacesContext context, UIComponent component, String value) throws ConverterException;*
  Devolver un nuevo objeto creado a partir de la cadena `value` invirtiendo la lógica aplicada en `getAsString()` de tal manera que se cumpla que:

  $$getAsString(context, component, getAsObject(context, component, value)) == value$$

- *String getAsString(FacesContext context, UIComponent component, Object value) throws ConverterException;*
  Devolver la cadena que representa al objeto `value` invirtiendo la lógica aplicada en `getAsObject()` de tal manera que se cumpla que:

  $$getAsObject(context, component, getAsString(context, component, value)) == value$$

Realizar estas transformaciones conviene mucho a la hora de convertir valores que son objetos, en valores serializables para su utilización en HTML y su posterior paso de parámetros vía HTTP, de forma que se pueda mapear objetos en unidades de información representables en texto imprimible que sean correctamente interpretables por el navegador. Esta conversión debería ser invertible para ser usada durante todo el flujo de la conversación cliente-servidor.

5.3.7. `com.autentia.intra.dao`

Las siguientes clases son básicas para implementar la capa de acceso a datos. Servirán para conectar la capa de modelo con la de persistencia.

5.3.7.1. Excepciones

Las excepciones definidas en este paquete son las siguientes:

- **DataAccException** Clase base para las dos siguientes:
- **DataIntegrityException** Utilizar cuando fallan las reglas de integridad de la base de datos, como la de unicidad o la de referencia.
- **DataNotFoundException** Utilizar cuando no se puede encontrar un objeto en la base de datos que cumpla con los criterios de búsqueda.

5.3.7.2. IDataAccessObject

Esta interfaz servirá de base para las clases de acceso a datos de cada modelo del sistema.

El contrato es el siguiente:

- *public interface IDataAccessObject<T extends ITransferObject>*
  Habrá que implementar una clase de acceso para cada clase de modelo, por eso la interfaz es `T`-paramétrica.
- *public T getById(int id) throws DataAccException*
  Recupera un objeto a partir de su `id`, que siempre es numérico.
- *public List<T>search(SortCriteria crit) throws DataAccException*
  Busca y devuelve todos los objetos que coinciden con los criterios de búsqueda que especifica `crit`. En caso de ser `null`, se devuelven todos los objetos existentes del mismo tipo.
- *public List<T>search(SearchCriteria search, SortCriteria sort) throws DataAccException*
  Igual que el anterior, pero además especificando el criterio de ordenación.
- *public void insert(T to) throws DataAccException*
  Guarda el objeto `to` en la base de datos.
- *public void update(T to) throws DataAccException*
  Actualiza el objeto, que previamente existe en la base de datos, con el estado interno del objeto actual `to`.
- *public void delete(T to) throws DataAccException*
  Elimina el objeto `to` de la base de datos.

5.3.8. `com.autentia.intra.dao.hibernate`

Todas estas clases se nombran como el modelo seguido de `DAO`. Por ejemplo: **EnsayoDAO** para la clase de acceso a datos del modelo de **Ensayo** correspondiente. Hereda la mayor parte de la funcionalidad de la clase abstracta **HibernateManagerBase**, e implementa la interfaz **IDataAccessObject<Ensayo>**, y así sucesivamente con el resto de clases.

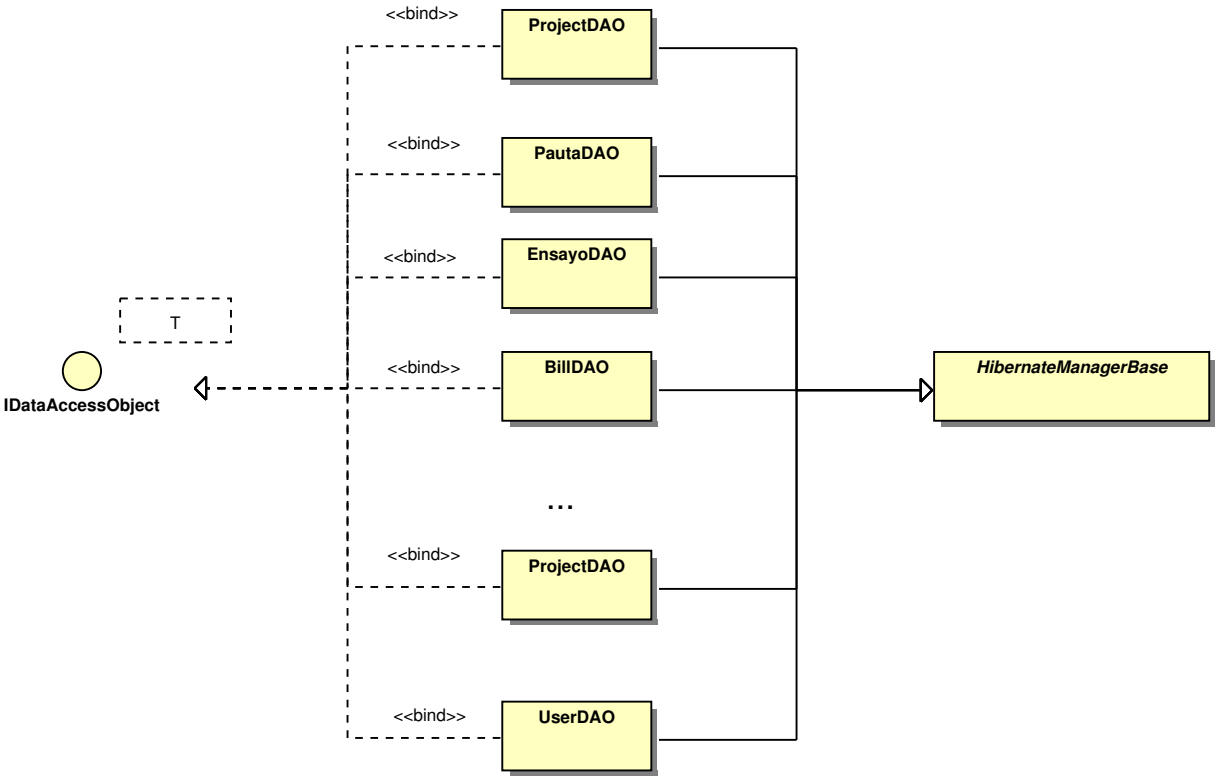

Figura 5.63: Diagrama de clases: com.autentia.intra.dao.hibernate

CAPÍTULO 6

Implementación

En este capítulo pasamos a explicar los detalles de implementación del sistema, y los problemas surgidos durante el desarrollo.

Empezamos localizando el código fuente de la aplicación base, tntconcept 0.9.1, descargable desde [2]. El código fuente se distribuye, bajo licencia libre GPLv2, en forma de dos ficheros:

`tntconcept-0.9.1-sources.jar`
: Se puede descomprimir como si se tratara de un fichero ZIP, y contiene los ficheros fuente en JAVA y algunos ficheros de recursos.

`tntconcept-0.9.1.pom`
: Es el descriptor del proyecto, necesario para el sistema de compilación automática de Maven.

No obstante, para poder producir un archivo objeto similar al que se incluye en el paquete de instalación hace falta organizarlo como especifica la jerarquía estándar para Maven, explicada en la sección 5.1.3; y para ello necesitamos extraer los ficheros de la aplicación J2EE, que se pueden obtener descomprimiéndolo como si fuera un archivo ZIP, el archivo de la aplicación web `tntconcept-0.9.1.war`. También de este archivo tuvimos que extraer el fichero `jsfcomponents-core-1.1.jar` e instalarlo en el repositorio local Maven ya que es una dependencia de compilación definida en el descriptor POM pero que sin embargo no se encuentra en ningún repositorio Maven público.

Para ello, debemos ejecutar el siguiente comando:

```
mvn install:install-file -Dfile=/ruta/al/jar -DgroupId=com.autentia \
-DartifactId=jsfcomponents-core -Dversion=1.1 -Dpackaging=jar
```

Con todo esto, ya podemos compilar la aplicación normalmente con `mvn package`, o probarla directamente con `mvn tomcat:run`.

La aplicación necesita que la base de datos se encuentre rellena con una serie de datos iniciales, que se consiguen importando al esquema configurado el contenido del fichero `createTables.sql`.

6.1. Modelo de datos

Al usar Hibernate como motor de persistencia, hay que especificar las propiedades de las clases que deseamos persistir. Vemos aquí un ejemplo de uno de los XML de configuración.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="com.autentia.intra.businessobject.Ensayo">
        <id name="id" type="java.lang.Integer">
            <generator class="native"/>
        </id>
        <property name="name" type="java.lang.String"/>
        <property name="nameIngles" type="java.lang.String"/>
        <set name="procedimientos" inverse="true">
            <one-to-many
            class="com.autentia.intra.businessobject.EnsayoProcedimiento"/>
        </set>
        <property name="description" type="java.lang.String"/>
        <property name="cost" type="java.math.BigDecimal"/>
        <property name="ownerId" type="java.lang.Integer"/>
        <property name="departmentId" type="java.lang.Integer"/>
        <property name="insertDate" type="java.util.Date"/>
        <property name="updateDate" type="java.util.Date"/>
        <property name="updatedById" type="java.lang.Integer"/>
        <property name="dimensional" type="boolean"/>
    </class>
</hibernate-mapping>
```

Como se observa, es un listado de las propiedades que contiene la clase **Ensayo** indicando su tipo de dato. Recordamos que, por ejemplo su propiedad **description** se define con el *setter*:

```
public void setDescription(String description) {
        this.description = description;
}
```

y el correspondiente *getter*:

```
1 public String getDescription() {
2         return this.description;
3 }
```

En las versiones más actuales de Hibernate, el tipo se deduce por introspección del código.

Para sincronizar el esquema de la base de datos a partir de la información de la configuración del mapeo, se emplea la herramienta que incluye el sistema, que se encuentra implementada en la clase `org.hibernate.tool.hbm2ddl.SchemaUpdate`, y se puede ejecutar desde la máquina virtual de Java.

6.2. Análisis del control de versiones

Para poder seguir y controlar los cambios al código, decidimos versionarlo con Subversion, que es bastante sencillo, sobre todo cuando se cuenta con la inestimable ayuda de [21]. El cliente SVN es bastante sencillo de usar, pero la gestión del repositorio en el lado servidor requiere un poco más de práctica.

Mostramos a continuación algunas estadísticas de la historia del código que se puede analizar con la herramienta StatSVN [26].

En la figura 6.1 podemos ver la distribución de ficheros de código en el repositorio clasificados por tipo y ordenados por el número de líneas de código (Lines Of Code (LOC)).

File Types

Type	Files	LOC	LOC per file
*.java	443 (30.6%)	132213 (46.0%)	298.4
*.jrxml	178 (12.3%)	73207 (25.5%)	411.2
*.jsp	265 (18.3%)	47803 (16.6%)	180.3
*.css	8 (0.6%)	9640 (3.4%)	1205.0
*.xml	84 (5.8%)	9141 (3.2%)	108.8
*.html	1 (0.1%)	4662 (1.6%)	4662.0
*.dump	1 (0.1%)	2533 (0.9%)	2533.0
*.properties	9 (0.6%)	2514 (0.9%)	279.3
*.sql	9 (0.6%)	1267 (0.4%)	140.7
*.tld	3 (0.2%)	1045 (0.4%)	348.3
Others	21 (1.5%)	3085 (1.1%)	146.9
Non-Code Files	424 (29.3%)	0 (0.0%)	0.0
Totals	1446 (100.0%)	287110 (100.0%)	198.5

Figura 6.1: Distribución de ficheros por tipo

El repositorio comenzamos a emplearlo para el mantenimiento de la aplicación y no durante las primeras fase del proyecto. En la figura 6.2 se puede observar el aumento de líneas de código durante la etapa de mantenimiento.

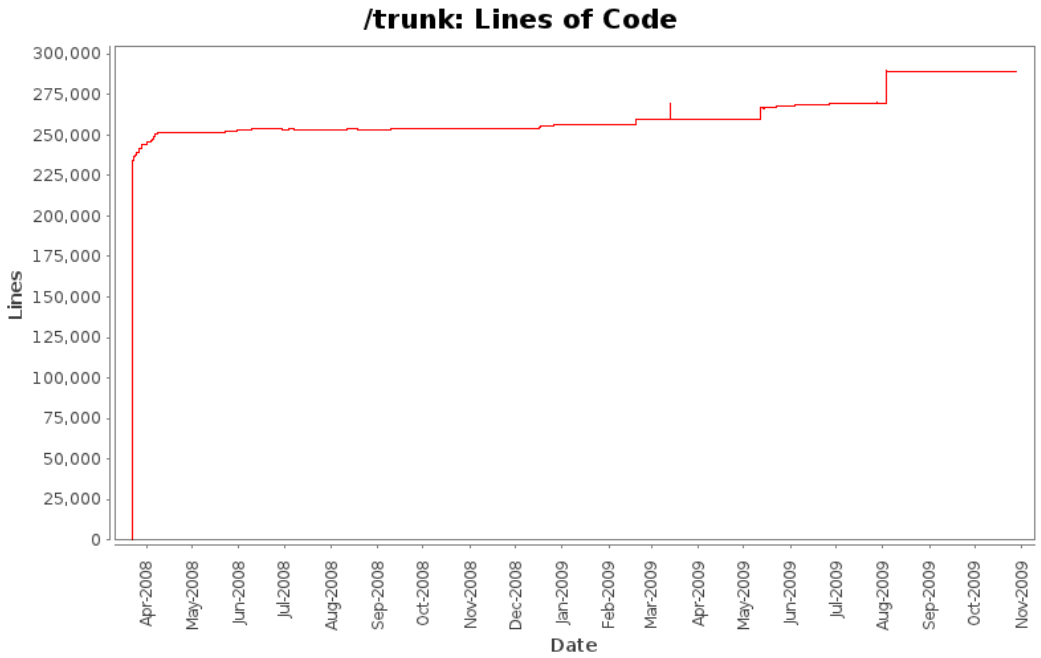

Figura 6.2: Evolución de LOC en `/trunk`

Como curiosidad, podemos ver las palabras que más se repiten como comentarios de *commits* en la nube de tags (figura 6.3).

Figura 6.3: Nube de tags en comentarios de commit

6.3. Herramientas de desarrollo

Para editar los ficheros fuente, empleamos el IDE (Entorno de Desarrollo Integrado) NetBeans [16] ahora de Oracle y antes propiedad de Sun. Abusamos especialmente de las características de depuración gráfica, autoformateado del código, y la integración con Subversion.

Para la elaboración de los informes que debe incluir la aplicación (ficheros `*.jrxml`), no se editaron manualmente, sino usando para ello un IDE basado en NetBeans que es el oficial de la biblioteca JasperReports, y que viene incluído con su distribución: *iReport* que se puede obtener de forma gratuita y libre desde [12].

Otra herramienta que empleamos fue Trac [29], como gestor para las incidencias que los usuarios nos iban haciendo llegar. En este caso, cada *commit* podría referirse a una incidencia concreta, para poder hacer un seguimiento del esfuerzo empleado en solucionar cada error o en implementar cada nueva funcionalidad solicitada, que son los dos tipos de incidencias que tenemos.

CAPÍTULO 7

Pruebas y validación

A lo largo de todo el ciclo de vida del proyecto, es indispensable asegurarse que se están siguiendo los pasos de forma adecuada, para detectar un fallo lo antes posible e iniciar las tareas necesarias para su resolución.

Cada fase del ciclo termina con el desarrollo de un producto que ha de ser comprobado y validado.

7.1. Verificación y validación del análisis

La fase de **análisis** concluye con documento de Especificación de Requisitos del Sistema (SRS), que es lo que se puede encontrar en el capítulo 4. Este documento hay que someterlo a dos comprobaciones:

a) **Verificación.** Una revisión interna de todos los requisitos que se especifican. Debemos comprobar que no existen requisitos contradictorios entre sí, a todos los niveles: de información, funcionales y no funcionales. Superada esta prueba, diremos que el software es *correcto*.[1]

b) **Validación.** Una vez determinado que la especificación de requisitos es correcta, hay que comprobar que es *válida*.[2] Es decir: *que satisface las necesidades del cliente.*

[1] «¿Se está construyendo correctamente el producto? *(Am I building the product right?)*», [3]
[2] «¿Se está construyendo el producto correcto? *(Am I building the right product?)*», [3]

7.1.1. Verificación del análisis

El proceso seguido para comprobar la correción de la especificación es simplemente contrastar cada requisito, con sus requisitos relacionados y ver que no se solapan (que tratan un aspecto individual) y que no son incoherentes.

7.1.2. Validación del análisis

Para probar que los requisitos son válidos, se entregan al cliente para su estudio y certificación. En caso negativo habría que corregir la especificación y repetir el proceso hasta que el cliente esté conforme. En este caso no fue necesario porque con bastante frecuencia se acordaban los requisitos finales: tras cada entrevista se realizaba un acta con los cambios solicitados con lo que se iba completando la especificación y validándola de forma progresiva.

7.2. Pruebas al sistema

Una vez cerrada la especificación, comenzamos a diseñar los cambios que se necesitan realizar al código (véase capítulo 5). Cada unidad de código que se implementa, se integra con el resto del sistema y se prueba realizando una compilación completa. Se despliega en el entorno de desarrollo y se realiza una prueba manual desde la interfaz de usuario. Si todo funciona como se esperaba en este requisito, podemos programar la prueba para que se pruebe automáticamente.

A lo largo de esta sección emplearemos el anglicismo *test* para referirnos a los *casos de prueba*.

Programar estas pruebas tiene las siguientes ventajas:

- Poder identificar un efecto colateral en una funcionalidad ya provada que se pueda producir al introducir una funcionalidad nueva.
- Detectar un fallo previamente solucionado que ha reaparecido, si programamos su *test de regresión* después de solucionar éste la primera vez.
- Ahorrar tiempo, si se quiere repetir las pruebas en diversos momentos del tiempo, en distintas fases del desarrollo, o integrado en distintos entornos.

Existen diversos *frameworks* para realizar las pruebas. En este proyecto vamos a usar Selenium IDE [25], que funciona como añadido para el navegador web Mozilla Firefox. Su uso es bastante sencillo: se puede crear un conjunto de pruebas (*test suite*) como una secuencia de pruebas (*tests*) individuales, cada una de las cuales es una secuencia de acciones realizadas sobre la página web. Se puede escribir cada instrucción en su lenguaje propio, llamado **selenese**, o bien se pueden capturar los eventos del ratón y el teclado que van generando el cuerpo del test.

De esta forma, los tests son secuencias de acciones realizadas sobre la aplicación, que tienen éxito (señalado con una marca verde) si en la repetición se comporta exactamente igual que cuando se grabó el test. En caso contrario, es decir, si se obtiene un resultado diferente a lo esperado para alguna de las acciones que componen el test, se dice que el test ha fallado (señalado en rojo).

Cuando un test falla, se procede de dos formas:

- Si el nuevo comportamiento es el correcto, actualizamos el código del test para reflejar la nueva situación.
- Si el anterior comportamiento era el correcto, tenemos que corregir el código de la aplicación hasta que el test se supere con éxito.

Puesto que tenemos el proyecto controlado bajo un sistema de control de versiones, podemos volver a una versión antigua del código para descubrir qué *commit* es el que produjo el fallo, como ayuda para localizar el punto de error que hay que corregir.

A continuación describimos cada una de las pruebas que se programaron. Los *tests* se ejecutan suponiendo que existe cargada en la base de datos una mínima información en ella. El fichero SQL con el volcado de prueba está incluído en la carpeta de proyecto, para poder comenzar la *suite* de tests con datos iniciales.

7.2.1. LoginFailed.html

Este test verifica lo siguiente:

1. Se limpian la caché y las cookies del navegador.
2. Se comprueba que la página inicial existe, y que tiene un formulario de entrada de usuario y contraseña.
3. Se introduce un nombre de usuario existente de prueba y una contraseña incorrecta.
4. Se envía el formulario.
5. Se comprueba que aparece el mensaje de error *Acceso no autorizado*.

7.2.2. LoginOK.html

Este test verifica lo siguiente:

1. Se limpian la caché y las cookies del navegador.
2. Se comprueba que la página inicial existe, y tiene un formulario de entrada de usuario y contraseña.
3. Se introduce un nombre de usuario existente de prueba y su contraseña correcta.
4. Se envía el formulario.
5. Se comprueba que aparece la pantalla con la barra de menú de opciones, y el tablón de anuncios.

Los subsiguientes tests, al ejecutarse en secuencia, presuponen que ya el usuario de prueba se encuentra logado.

7.2.3. OrganizationBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Contactos » Organizaciones*, se hace clic en ésta y se comprueba que se muestra el listado de empresas. Una de ellas deberá llamarse «Nuestra empresa».
2. Se pulsa en el botón de crear nueva organización y se comprueba que se muestra un formulario de entrada de datos.
3. Se pulsa en el botón de guardar, y se comprueba que aparece un error de validación por no haber rellenado el campo obligatorio *nombre*.
4. Se rellena el campo nombre con un valor de prueba, se comprueba que reaparece el listado de organizaciones, pero existe ahora una nueva fila para la organización recién creada, para probar que se ha creado correctamente el objeto.
5. Se pulsa sobre la fila para editar la recién creada organización, se introduce una coletilla al nombre, se envía el formulario, y se comprueba que en el listado de organizaciones que reaparece se muestra el nombre nuevo de la organización modificada, para probar que se ha actualizado correctamente el objeto.
6. Se pulsa sobre el botón de duplicar organización, se comprueba que aparece el formulario de edición del duplicado, se envía el formulario y se comprueba que en el listado que reaparece existe una nueva fila para el duplicado, para comprobar la funcionalidad de duplicar organizaciones.
7. Se pulsa sobre la última organización creada y se elimina, confirmando su borrado; se pulsa sobre la penúltima organización creada y se elimina, confirmando su borrado; se comprueba que en el listado no aparecen éstas, para probar la eliminación de objetos y dejar la base de datos en el mismo estado.

7.2.4. ContactBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Contactos » Contactos*, se hace clic en ésta y se comprueba que aparece el buscador de contactos. Se envía el formulario de búsqueda, vacío de criterios. Se comprueba que aparece el listado de contactos, y que uno de ellos deberá llamarse «Luis Cabeza Ruiz», como está en los datos iniciales.
2. Se pulsa en el botón de crear nuevo contacto y se comprueba que aparece un formulario de creación de contactos. Se rellena el campo nombre con un valor de prueba, se envía el formulario y se comprueba que de nuevo aparece el listado de organizaciones, pero que ahora existe una nueva fila para el contacto recién creado, para probar que se ha creado correctamente el objeto.

3. Se pulsa sobre la fila para editar el recién creado contacto, se introduce una coletilla al nombre, se envía el formulario, y se comprueba que en el listado aparece, existe el nombre nuevo, para probar que se ha modificado correctamente el objeto.

4. Se pulsa sobre el último contacto creado y se comprueba que aparece el formulario de edición. Se pulsa sobre el botón de eliminar, confirmando su borrado; se comprueba que en el listado que aparece no existe el contacto recién creado, para probar la eliminación de objetos.

7.2.5. OfferBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Contactos » Presupuestos*, se hace clic en ésta y se comprueba que aparece el buscador de presupuestos. Se envía el formulario vacío de criterios, y se comprueba que aparece un listado de presupuestos.

2. Se pulsa en el botón de crear nuevo presupuesto, se comprueba que aparece el formulario de creación de presupuesto. Se rellenan algunos campos, y se envía el formulario. Se comprueba que reaparece el formulario de edición y entonces se pulsa sobre el botón de regresar al listado. Se comprueba que en el listado que se muestra aparece una nueva fila con el código del presupuesto recién creado, para probar la creación de presupuestos.

3. Se pulsa en la fila del presupuesto y se comprueba que aparece el formulario de edición, se cambia el campo título y se envía, confirmando la modificación. En el listado que aparece, se comprueba que existe la fila con el título cambiado, para probar la modificación de presupuestos.

4. Se pulsa en la fila del presupuesto, se comprueba que aparece el formulario de edición, se pulsa en el botón de eliminar y se confirma el borrado. En el listado que aparece, se comprueba que no existe fila con el título del presupuesto que se acaba de borrar, para probar la eliminación de presupuestos.

7.2.6. PautaBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Tablas maestras » Pautas*, se hace clic en ésta y se comprueba que se muestra el buscador de pautas. Se envía el formulario vacío de criterios, y se comprueba que muestra una listado de pautas.

2. Se pulsa en el botón de crear nueva pauta y se comprueba que aparece el formulario de creación de pautas. Se rellenan algunos campos, y se envía el formulario. Se comprueba que reaparece el formulario de edición y entonces se pulsa sobre el botón de regresar al listado. En el listado, se comprueba que aparece una fila con la nueva pauta.

3. Se pulsa en la fila de la pauta y se comprueba que aparece el formulario de edición. Se modifica el nombre, se envía el formulario, confirmando la actualización. Se comprueba que reaparece el formulario de edición y se pulsa en el botón de regresar al listado. Se comprueba en el listado que aparece que se muestra el objeto con el nombre nuevo.

4. Se pulsa en la fila de la pauta y se comprueba que aparece el formulario de edición. Se pulsa sobre el botón de eliminar y se confirma el borrado. En el listado que aparece, se comprueba que no existe una fila con el nombre de la pauta recién borrada.

7.2.7. EnsayoBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Tablas maestras » Ensayos*, se hace clic en ésta y se comprueba que aparece el buscador de ensayos. Se envía el formulario de búsqueda vacío de criterios y se comprueba que aparece el listado de ensayos.

2. Se pulsa en el botón de crear nuevo ensayo y se comprueba que aparece el formulario de creación de ensayos en blanco. Se rellena algunos campos y se envía el formulario. Se comprueba que reaparece el formulario de entrada de datos. Se pulsa en el botón de regresar al listado y se comprueba que aparece el ensayo recién creado en este listado.

3. Se pulsa en la fila del ensayo y se comprueba que aparece el formulario de edición del ensayo. Se modifica el nombre del ensayo y se envía el formulario, confirmando la actualización. Se comprueba que reaparece el formulario de edición, y se pulsa sobre el botón de regresar al listado de ensayos. Se comprueba que aparece el ensayo con el nuevo nombre.

4. Se pulsa en la fila del ensayo y se comprueba que aparece el formulario de edición del ensayo. Se pulsa sobre el botón de eliminar y se confirma el borrado. Se comprueba que aparece el listado de ensayos, pero sin la fila del ensayo que se acaba de eliminar.

7.2.8. NotaSalidaBean.html

Este test verifica lo siguiente:

1. Se comprueba que existe una opción del menú *Contactos » Notas de salida*, se hace clic en ésta y se comprueba que aparece el listado de notas de salida.

2. Se pulsa en el botón de crear nueva nota de salida y se comprueba que aparece el formulario de creación de notas de salida en blanco. Se rellena algunos campos y se envía el formulario. Se comprueba que reaparece el formulario de edición de notas de salida. Se pulsa en el botón de regresar al listado y se comprueba que aparece la nota de salida recién creada en este listado.

3. Se pulsa en la fila de la nota de salida y se comprueba que aparece el formulario de edición de la nota de salida. Se pulsa sobre el botón de imprimir nota de salida en PDF y se comprueba que se descarga un fichero PDF con la nota de salida. Se pulsa en el botón de regresar al listado y se comprueba que aparece el listado de notas de salida.

4. Se pulsa el botón de búsqueda y se comprueba que aparece el formulario de búsqueda de notas de salida. Se introduce el nombre de la nota de salida recién creada en el campo nombre y se envía el formulario. Se comprueba que en el listado que aparece existe la nota de salida buscada.

5. Se pulsa en la fila de la nota de salida y se comprueba que aparece el formulario de edición de la nota de salida. Se pulsa sobre el botón de eliminar y se confirma el borrado. Se comprueba que aparece el listado de notas de salida, pero sin la fila de la nota de salida que se acaba de eliminar.

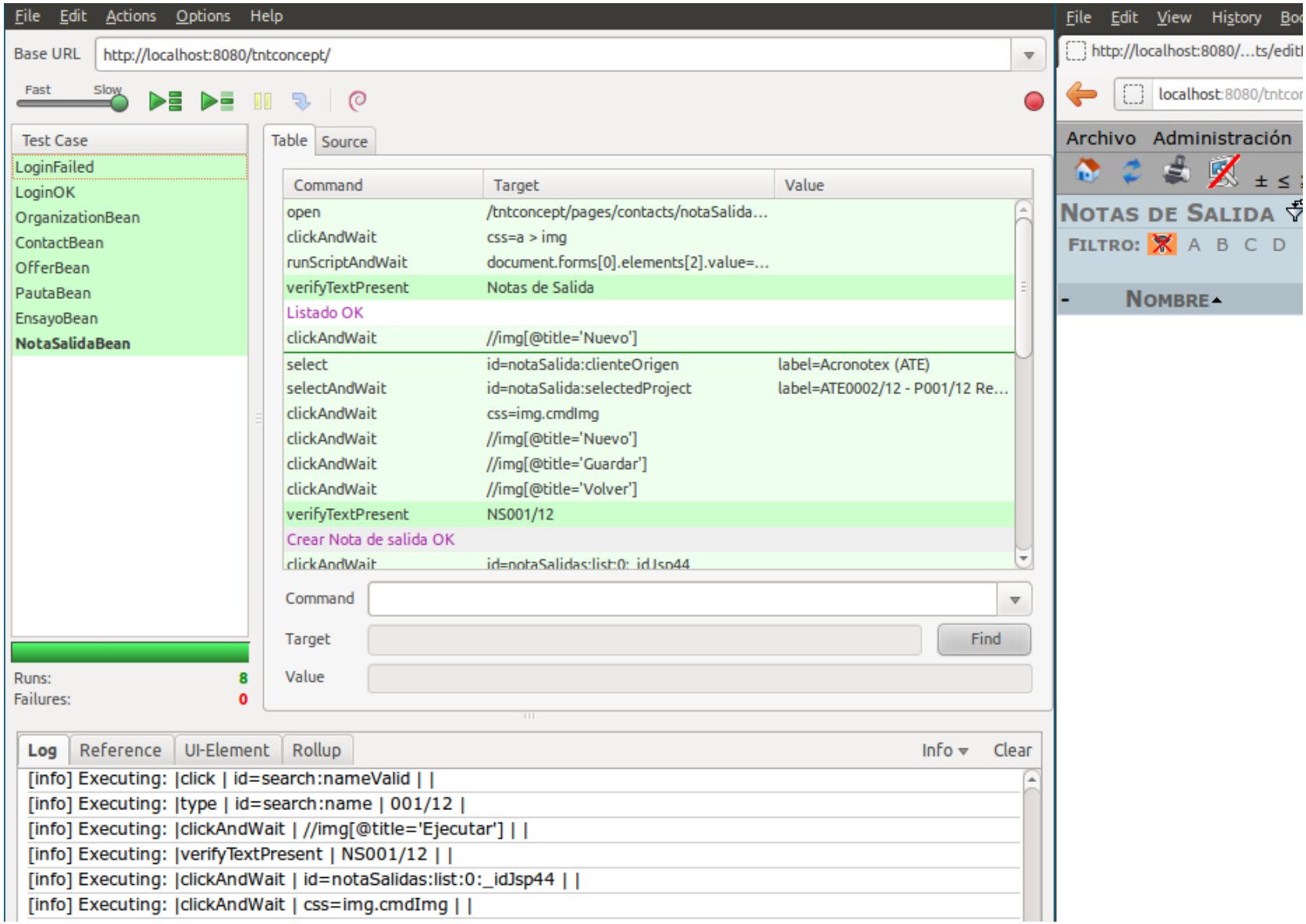

Figura 7.1: Ejecución de las pruebas en Selenium IDE

CAPÍTULO 8

Resumen

En este proyecto se adapta un sistema libre de gestión empresarial a una empresa cuya actividad principal es realizar ensayos a materiales industriales y pinturas. Este sistema permitirá incrementar la productividad de la empresa, al sustituir las herramientas ofimáticas por herramientas específicas de gestión empresarial, entre las que destaca TNTConcept. Además, en este proyecto se combinan las siguientes tecnologías: J2EE, JSF, JasperReports, Hibernate y MySQL.

8.1. Introducción y objetivos del proyecto

Este Proyecto se inició para resolver las necesidades de gestión interna de la empresa Titania, S.L.U. [28], cuya ocupación es la de realizar ensayos a materiales industriales y pinturas. Después de realizar los ensayos necesarios, se emite un informe con los resultados obtenidos y su conformidad según una serie de estándares del sector.

El sistema original estaba compuesto por un conjunto de hojas de cálculo de MS Excel junto con una base de datos en MS Access, que los técnicos de la empresa ya usaban, por lo que se asumen conocimientos básicos de informática y de ofimática.

El objetivo del proyecto es incrementar la productividad de la empresa realizando para ello un sistema de información que incorpore todo el ciclo de trabajo bajo la misma herramienta, adaptando un software de Planificación de Recursos Empresarial (ERP) existente de software fuente libre como lo es TNTConcept.

Al concluir el desarrollo, la aplicación web debe ser capaz de manejar clientes, productos (ensayos y pautas), pedidos, proyectos (informes de los ensayos realizados) y facturas.

8.2. Análisis del sistema

Para conocer exactamente los requerimientos de la aplicación, tuvimos varias reuniones con el cliente durante la primera semana de desarrollo. De estas reuniones pudimos concluir la Especificación de Requisitos del Software (SRS) [11] cuyos aspectos más relevantes resumimos a continuación.

8.2.1. Características de los usuarios

Los usuarios son empleados que conocen el funcionamiento de la empresa, y que además tienen conocimientos básicos del uso de un ordenador, navegador web, y aplicaciones ofimáticas de entornos Windows.

Identificamos tres actores que participan en nuestro sistema:

1. **Usuario.** Se refiere a una persona que se encuentra registrada en la aplicación y que puede conectarse e interactuar con el sistema. Es el actor base de las especializaciones *Administrador* y *Técnico*.
2. **Administrador.** Es un tipo de *Usuario* que además posee privilegios para acceder a todas las secciones de la aplicación y pueden modificar a otros *Usuarios*. Son los directores de departamentos y gestores de la empresa, con capacidad para efectuar cualquier modificación e inspección.
3. **Técnico.** Es un tipo de *Usuario* con acceso a las secciones de trabajo normal: productos, clientes, ensayos, informes, albaranes, facturas.

8.2.2. Otras restricciones

El sistema tendrá que funcionar en un servidor Windows® 2003 Server con 2 GiB de memoria RAM. Se podrán extraer copias de seguridad de los datos. Y la utilización simultánea por parte de una media de 10 usuarios conectados por la red local no puede generar retardos.

8.2.3. Requisitos de interfaces externas

8.2.3.1. Interfaz con el usuario

El usuario interactuará con el sistema a través de un navegador web en un entorno Windows. La interfaz gráfica de usuario tendrá un aspecto homogéneo en forma de tablas paginadas y formularios de edición. También se pueden generar archivos PDF descargables para imprimir.

8.2.3.2. Interfaz con el hardware

Las copias de seguridad programadas semanalmente de los datos de la aplicación se extraerán a una unidad de disco magneto-óptica de hasta 250 MB.

8.2.3.3. Interfaz con el software

La aplicación funcionará en Windows® 2003 Server, mientras que la vista del cliente se interpretará en un navegador Internet Explorer 6 en Windows XP o superior.

8.2.4. Objetivos

Resumiendo, el sistema debería ser capaz de:

- Gestionar **usuarios**, controlando sus permisos de acceso a las distintas áreas de la aplicación.
- Gestionar **clientes y proveedores**.
- Gestionar los **productos** que se ofrecen, con sus catálogos de ensayos, pautas y precios.
- Gestionar los **trabajos** que se realicen, compuesto por todos o algunos de los subsistemas:
  - Gestión de **presupuestos**.
  - Gestión de **informes**.
  - Gestión de **albaranes**.
  - Gestión de **facturas**.

8.2.5. Requisitos de información

El sistema almacenará la siguiente información relativa de las siguientes entidades:

Usuarios Datos personales como el nombre y apellidos, dirección, fecha de nacimiento. Y datos del usuario como nombre de usuario y contraseña, rol y departamento al que pertenece.

Departamentos Nombre del departamento, descripción, y departamento que lo contiene, si es un *sub-departamento*.

Clientes Nombre de la empresa, CIF, dirección postal, dirección de envío de informes, dirección de facturación, teléfonos y email de contacto.

Contactos Empresa y cargo, nombre completo, teléfonos y email de contacto.

Ensayos Nombre del ensayo (y en inglés), descripción, precio, procedimientos aplicables.

Procedimientos Ensayo aplicable, denominación oficial del procedimiento.

Pautas Nombre de la pauta, descripción, título del informe que genera, familia de pautas, identificadores del cliente, conjunto de ensayos y procedimientos incluídos en la pauta, precio.

8.3. Diseño del sistema

Para diseñar nuestro sistema, tenemos que tomar algunas decisiones básicas, como la tecnología a emplear y el/los lenguajes de programación a utilizar.

Partimos de la aplicación de software libre TNTConcept [2], que tenemos que modificar y adaptar para que el sistema resultante cumpla con nuestra especificación de requisitos obtenido en la etapa de análisis. Por este motivo, mantendremos el uso principalmente del lenguaje de programación orientado a objetos Java [7] para la codificación; usando además J2EE [5], JSF [24], Hibernate, MySQL, JasperReports [10] y Tomcat.

8.3.1. Arquitectura

La arquitectura que emplearemos es el de dos capas: cliente/servidor, aplicando el patrón de diseño en tres capas Modelo-Vista-Controlador (MVC, [8]), que separa aspectos diferenciados de la aplicación, a nivel del código:

Modelo Clases necesarias para el acceso a los datos.

Vista Clases y ficheros que componen la interfaz gráfica para la visualización de la información.

Controlador Clases que coordinan el paso de información entre vistas y modelos.

Los modelos usan el motor de persistencia objeto-relacional (ORM) ampliamente conocido y usado en aplicaciones J2EE llamado Hibernate. Esta biblioteca es la que, en última instancia, ejecuta las órdenes SQL de lectura y escritura en la base de datos de MySQL.

Las vistas son los ficheros de recursos web contenidos en la carpeta `webapp`, que utilizan las clases de los componentes de Apache MyFaces [30], que es la implementación que usamos del estándar JSF.

8.3.1.1. Diagrama de componentes

Podemos ver el diagrama de componentes en la figura 8.1.

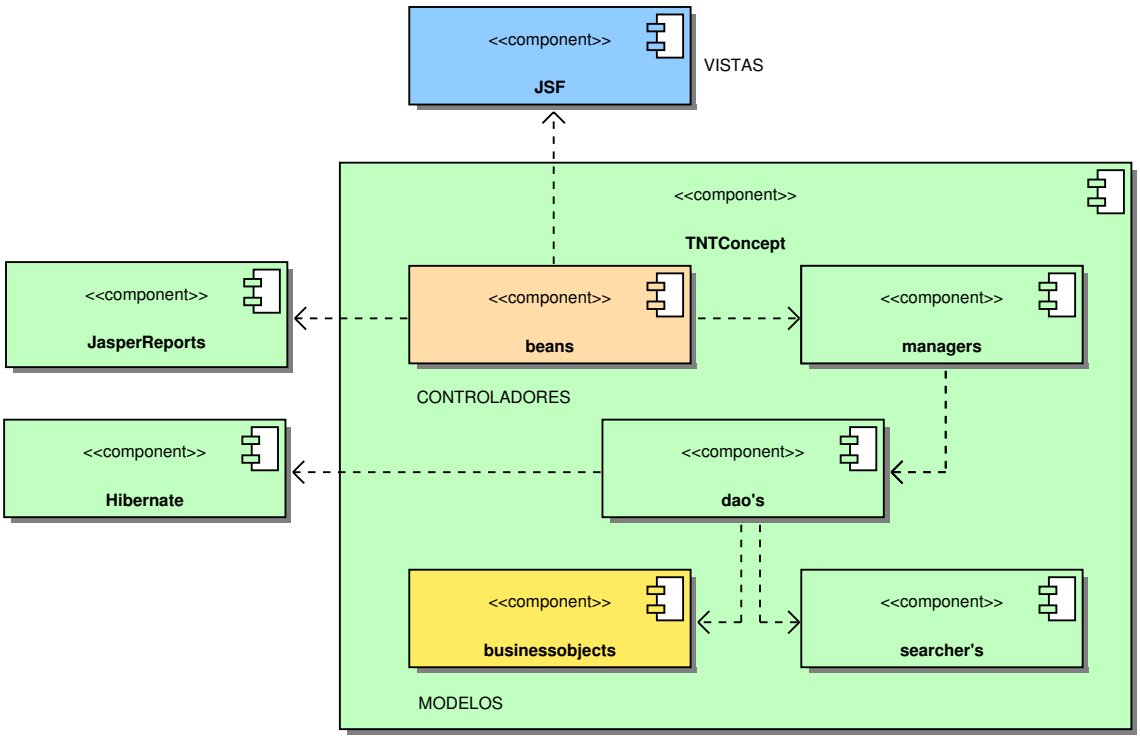

Figura 8.1: Diagrama de componentes software

Cada componente representa una agrupación por funcionalidades de las clases y sus ficheros de recursos. El componente *vista* del patrón MVC queda definido por el uso del *framework* JSF y las plantillas de la aplicación; los *controladores* son las clases que forman el paquete `beans`, y los *modelos* las clases del paquete `businessobjects`. El resto son las bibliotecas e interfaces en las que se apoyan controladores y modelos.

8.3.2. Entorno de ejecución

A continuación podemos ver el diagrama de despliegue de la puesta en producción (figura 8.2). Interviene un servidor y uno o más clientes, puesto que se ha definido una arquitectura para el sistema de cliente/servidor. El sistema servidor se compone del servidor de aplicaciones J2EE Tomcat 6 y del sistema de gestión de bases de datos MySQL 5. En el servidor de aplicaciones se desplegará un artefacto de aplicación web `tntconcept.war` en el contexto denominado `/tntconcept`, y en el SGBD se creará un usuario y una base de datos para esta aplicación en exclusiva.

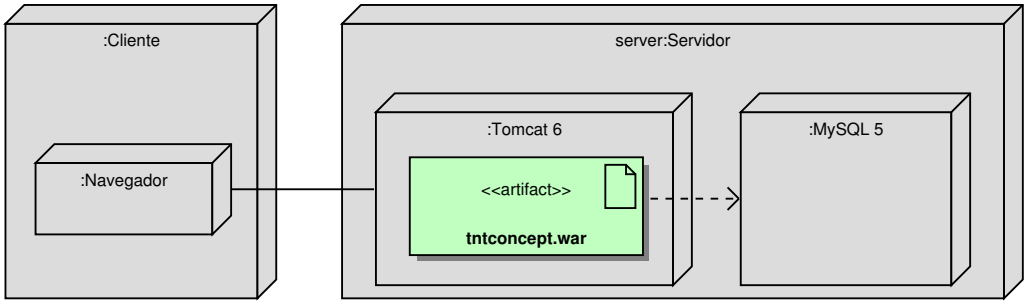

Figura 8.2: Diagrama de despliegue: Entorno de producción

8.3.3. Diagramas de clases

Mostramos a continuación algunos de los diagramas de clases en notación UML [13].

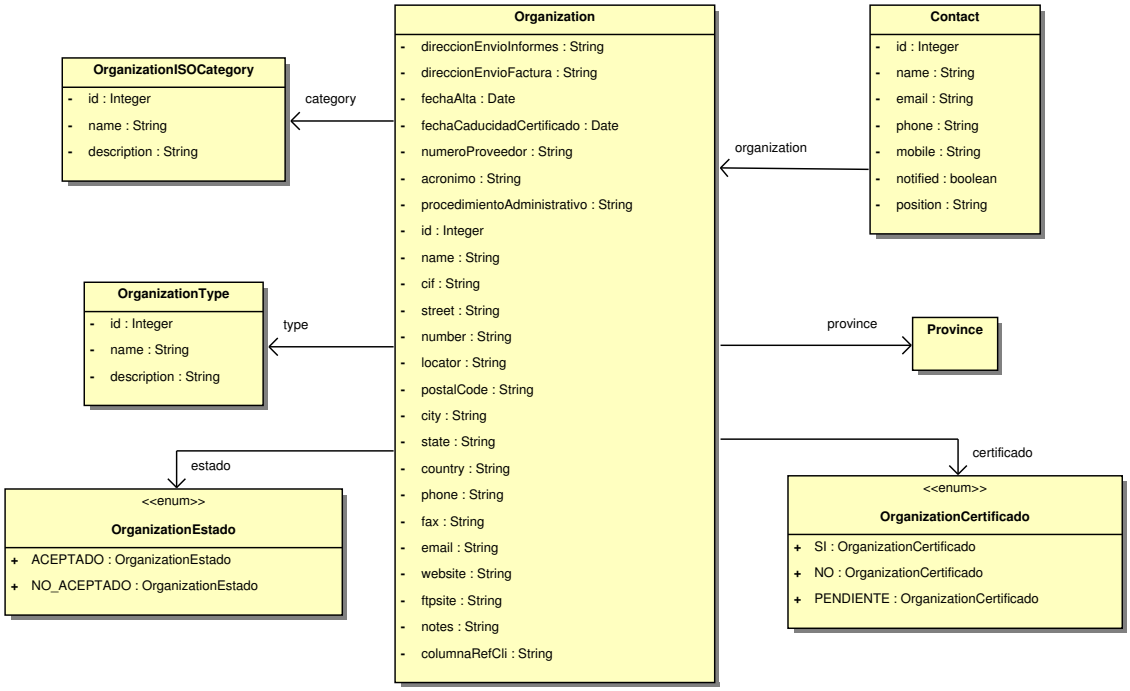

Figura 8.3: Diagrama de clases: Clientes

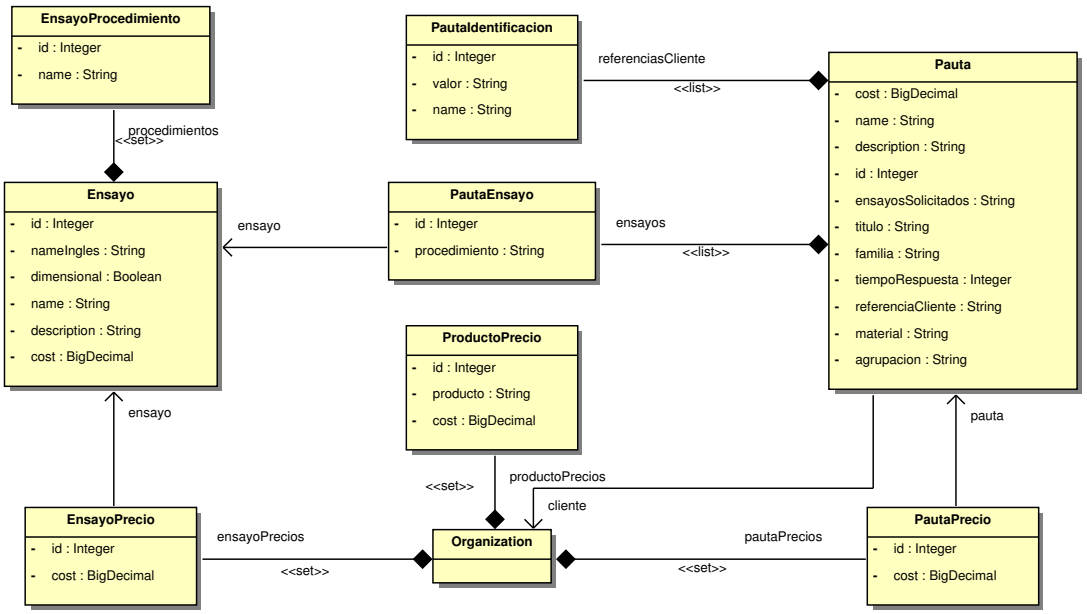

Figura 8.4: Diagrama de clases: Productos

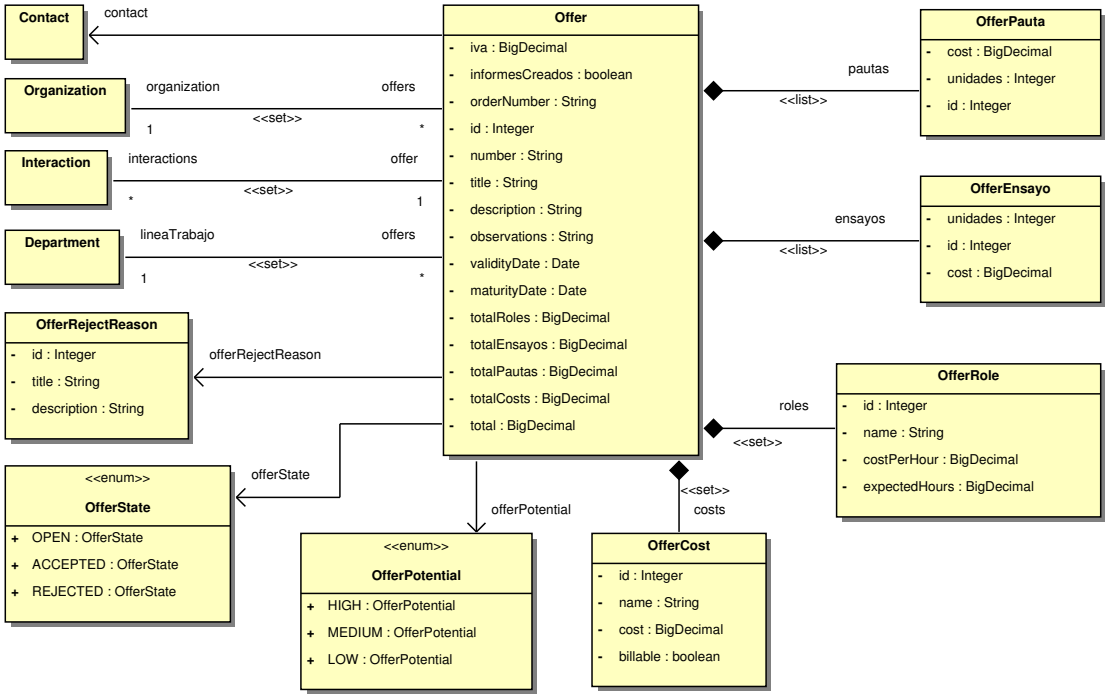

Figura 8.5: Diagrama de clases: Presupuestos

8.3.4. Paquetes

A continuación describimos los paquetes más importantes de la aplicación.

8.3.4.1. `com.autentia.intra.bean.*`

Las clases (*beans*) contenidas en este paquete y subpaquetes forman las acciones de los controladores en el patrón MVC. Todas ellas deben aceptar el constructor por defecto (sin argumentos), y tener *getters* y *setters* en *camelCase* para todas sus propiedades, de manera que las vistas de cada controlador puedan tener acceso a estas propiedades.

El resto de métodos públicos serán las funciones que manejan cada uno de los eventos producidos en la interfaz de usuario. Éstas efectúan las operaciones necesarias y devuelven `null` si la siguiente vista a renderizar es ella misma; o un objeto `String` indicando el *outcome* (una clave que relaciona cada vista `.jsf` con la siguiente, siguiendo la configuración de navegación del fichero `faces-navigation.xml`).

Cada uno de estos beans están más o menos normalizados de forma que cada uno sirve para gestionar un tipo de entidad, con funciones y vistas para crear un objeto nuevo, modificarlo, listar objetos existentes, realizar búsquedas de objetos, y borrarlos.

8.3.4.2. `com.autentia.intra.businessobject`

Este paquete aloja todas las clases que se deducen de los diagramas de clases, que son las que se conocen como *clases de dominio* del problema.

Estas clases sólo tienen atributos y sus métodos para acceder a ellos como si fueran propiedades, y son las que forman la capa modelo del patrón MVC.

8.3.4.3. `com.autentia.intra.businessobject.config`

Este paquete contiene la definición, un fichero XML por cada clase, de los modelos para el motor de persistencia Hibernate.

8.3.4.4. `com.autentia.intra.dao.hibernate`

Todas estas clases se nombran como el modelo seguido de `DAO`. Por ejemplo: **EnsayoDAO** para la clase de acceso a datos del modelo de **Ensayo** correspondiente. Hereda la mayor parte de la funcionalidad de la clase abstracta **HibernateManagerBase**, e implementa la interfaz **IDataAccessObject<Ensayo>**, y así sucesivamente con el resto de clases.

8.4. Implementación

Para implementar el sistema, hemos codificado usando lenguaje Java, HTML y JavaScript.

Hemos usado algunas herramientas que facilitan el ciclo de desarrollo, como son:

- El Entorno de Desarrollo Integrado (IDE) NetBeans.
- El sistema de compilación automática Maven [17].
- El sistema de control de versiones Subversion [21].
- El editor de plantillas de informes `*.jrxml` iReport [10].
- El gestor de incidencias Trac.
- El GUI para MySQL Query Browser.

8.5. Pruebas y validación

Hemos, en la fase de análisis, primero *verificado* [3] que los requisitos identificados son correctos, no se contradicen entre sí, son completos y poco acoplados. Y después hemos *validado* la especificación de requisitos contrastándola con el cliente hasta que nos diera su aprobación.

Las pruebas efectuadas sobre el entorno de desarrollo han sido probando la aplicación desde el navegador de forma interactiva, comprobando cada funcionalidad implementada con su comportamiento especificado. Estas pruebas se capturaron con Selenium IDE para poder repetirlas en cualquier momento que se desee comprobar que el funcionamiento del sistemas sigue siendo el correcto. Durante la fase de mantenimiento, el cliente por su cuenta hacía pruebas al sistema en un entorno de pruebas antes de traspasar los cambios al entorno de producción.

8.6. Conclusiones y trabajo futuro

Este proyecto ha servido para solucionar el problema de gestión interna de una empresa de realización de ensayos a materiales. Hemos realizado una especificación de requisitos que ha validado el cliente. Después hemos analizado software libre existente para adaptar a estas necesidades, diseñando los cambios e implementándolos en el entorno de producción.

Una de las posibles ampliaciones de este sistema podría ser la realización de un par de frontales para *tablets*, una aplicación iOS y otra Android, que se conecten al sistema mediante un servicio web para la transferencia de datos. De esta forma, los técnicos pueden completar los resultados obtenidos en el mismo lugar de los ensayos sin necesidad de acudir a un ordenador. Al ser software libre, cualquier persona de la comunidad podría hacer ésta u otras modificaciones sin ningún problema.

CAPÍTULO 9

Conclusiones y trabajo futuro

Concluyendo, podemos decir que toda la ejecución de este proyecto ha servido para crear una solución al tradicional problema de gestión empresarial: se dispone de mucha información pero ésta no se encuentra organizada de una forma óptima que permita un manejo eficaz. Esta clase de necesidades son bastante frecuentes en las empresas, y prácticamente cualquier programa de gestión se parece a cualquier otro y podría servir para cubrir los requisitos básicos, entre los que se encuentran la gestión de la información sobre de clientes, pedidos y facturas.

No obstante, no sería lógico crear una herramienta (en principio, para que nos facilite nuestra tarea, y aumentar la productividad) para tener después que adaptar el funcionamiento de la empresa a esta herramienta. Si no más bien todo lo contrario, es el software el que debe adaptarse a las necesidades del usuario.

Y esto es lo que favorece el software libre: la personalización del mismo para lograr una conexión perfecta entre el sistema informático y sus usuarios, con todas las demás sabidas ventajas de la reutilización, abaratamiento de costes y reducción de los tiempos de desarrollo.

Personalmente, la realización de este proyecto me ha aportado bastantes conocimientos sobre las tecnologías y herramientas que paso a enumerar a continuación.

- Tecnologías de desarrollo: como el lenguaje de programación *Java*, la forma de usar *Hibernate* como ORM, *JasperReports* como generador de informes, la programación de interfaces web orientadas a componentes, para ello usando la especificación *JSP* y el framework de desarrollo *JSF*. La instalación y configuración del servidor de aplicaciones *Apache Tomcat* y del Sistema de Gestión de Bases de Datos (SGBD) *MySQL*. También he aprendido cómo aplicar correctamente el patrón de diseño Model-View-Controller (MVC).

- Herramientas de desarrollo: como los IDE *Netbeans* e *IntelliJ IDEA*, el gestor de proyecto *Maven 2*, los sistemas de control de versiones *Subversion* y *Git* y el diseñador de informes *iReport*. También he aprendido a manejar el gestor de proyecto y seguimiento de errores *Trac*.
- Herramientas de documentación: como el sistema de composición tipográfica LaTeX2e para realizar la memoria y el editor de diagramas UML *BOUML*. También he usado el planificador de proyectos *Planner* [22] para elaborar el diagrama de Gantt.

Todos los proyectos están en continua evolución mientras se encuentren en uso, y mientras sus usuarios demanden más funcionalidades. En éste, se podría introducir la siguiente mejora:

- Versión para *tablets*.

  Las tabletas están teniendo un auge importante en los últimos meses. Se podría hacer una versión para el sistema iOS y otra para Android, con la que los técnicos pudieran completar los datos del informe de una forma más cómoda, desde el lugar donde se está realizando el ensayo o la medición, sin necesidad de tener un ordenador cerca. Habría que desarrollar las dos aplicaciones y el servicio web que hiciera de capa de interoperabilidad para la transferencia de los datos de forma centralizada en el mismo sistema.

APÉNDICE A

Manual de instalación

Suponemos que el sistema es Linux, Fedora 14, por ejemplo, pero el procedimiento es fácilmente exportable a cualquier otra distribución.

Para instalar los paquetes requeridos, sigamos el procedimiento habitual para la instalación de paquetes:

```
1 # yum install java-1.6.0-openjdk-devel mysql-server mysql \
2 > mysql-connector-java tomcat6-admin-webapps tomcat6 tomcat-native
```

El siguiente paso es cargar en la base de datos la estructura y datos iniciales de la aplicación. Creamos la base de datos, y a continuación cargamos el script especificando la ruta al archivo `createTables.sql` que contiene la estructura de datos inicial:

```
$ mysql -uroot
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.1.52 Source distribution

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database tnt;
Query OK, 1 row affected (0.00 sec)

mysql> use tnt;
Database changed

mysql> source installer/sql/mysql/createTables.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.12 sec)

...

Query OK, 0 rows affected (0.12 sec)

mysql> quit
Bye
```

A continuación tendremos que copiar el archivo `tntconcept.war` en la carpeta `$TOMCAT_HOME/webapps`. En el caso de Fedora, ésta es:

```
# install -m644 tntconcept.war /usr/share/tomcat6/webapps/
```

El siguiente paso es configurar el contexto de despliegue. Creamos un nuevo archivo `/etc/tomcat6/Catalina/localhost/tntconcept.xml` con el siguiente contenido:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/tntconcept" docBase="tntconcept">
  <Manager pathname=""/>
  <Environment name="TNTConceptConfigDir"
      type="java.lang.String" value="/etc/tntconcept"/>
  <Resource name="jdbc/TNTConcept" auth="Container"
      type="javax.sql.DataSource"
      maxActive="100" maxIdle="30" maxWait="10000"
      driverClassName="com.mysql.jdbc.Driver"
      username="root" password=""
      url="jdbc:mysql://localhost:3306/tnt?autoReconnect=true"/>
</Context>
```

Incluir el driver de MySQL para JDBC en el path de tomcat así (lo ponemos como un enlace simbólico, aunque también se podría copiar directamente el archivo `.jar`):

```
# ln -fs /usr/share/java/mysql-connector-java.jar \
> /usr/share/tomcat6/lib/
```

Una vez configurado el servidor de tomcat para correr la aplicación; ésta no funcionará hasta no satisfacer su propia configuración.

Empezamos copiando el esqueleto de configuración y después modificamos los valores necesarios:

```
# install -m755 -d /etc/tntconcept
# cp -rfa --no-preserve=ownership installer/config/* /etc/tntconcept/
```

Editamos el fichero `/etc/tntconcept/autentia.properties` descomentando las líneas que preceden con `# Unix version`

```
...
pathFicheros=/var/lib/tntconcept/upload
...
pathReports=/var/lib/tntconcept/reports
...
```

Editamos el fichero `/etc/tntconcept/log4j.properties` de la misma forma:

```
...
log4j.appender.file.File=/var/log/tntconcept/intraweb.log
...
log4j.appender.migration.File=/var/log/tntconcept/migration.log
...
log4j.appender.security.File=/var/log/tntconcept/security.log
...
```

Creamos los 3 directorios necesarios, con permisos para el usuario que ejecute el servidor tomcat:

```
# install -m755 -d /var/lib/tntconcept/upload
# install -m755 -d /var/lib/tntconcept/reports
# install -m755 -d /var/log/tntconcept
# chown tomcat6:tomcat6 /var/lib/tntconcept/upload
# chown tomcat6:tomcat6 /var/lib/tntconcept/reports
# chown tomcat6:tomcat6 /var/log/tntconcept
```

Una vez instalado, la primera ejecución necesita la migración de los datos (de fábrica) a través de las distintas versiones del esquema de base de datos que ha ido evolucionando con el programa.

Para ello, primero reiniciamos el servidor para hacer efectivos todos estos cambios:

```
# service tomcat6 restart
```

Consultamos en el navegador la dirección de la aplicación:

`http://localhost:8080/tntconcept/`

En la pantalla emergente que aparece de login (figura A.1), escribir las credenciales por defecto, que son:

- **Usuario**: `admin`
- **Contraseña**: `adminadmin`

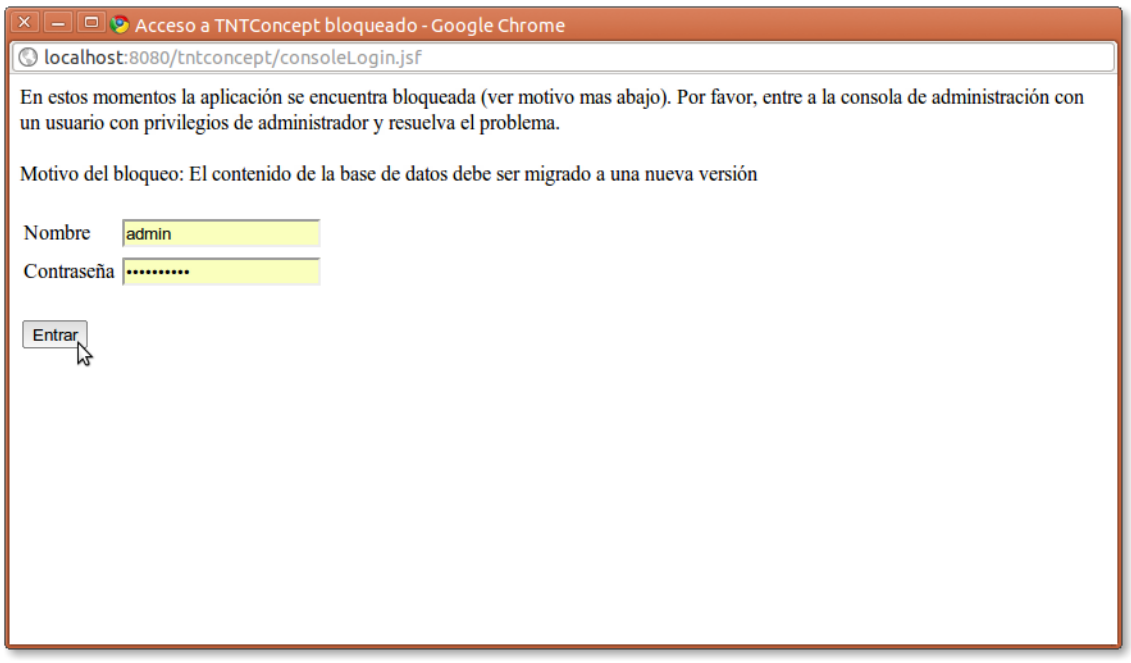

Figura A.1: Acceso a la consola de migración

Después del mensaje de advertencia, hacer clic sobre «Migrar base de datos», y después sobre el botón, para continuar con la instalación (figura A.2).

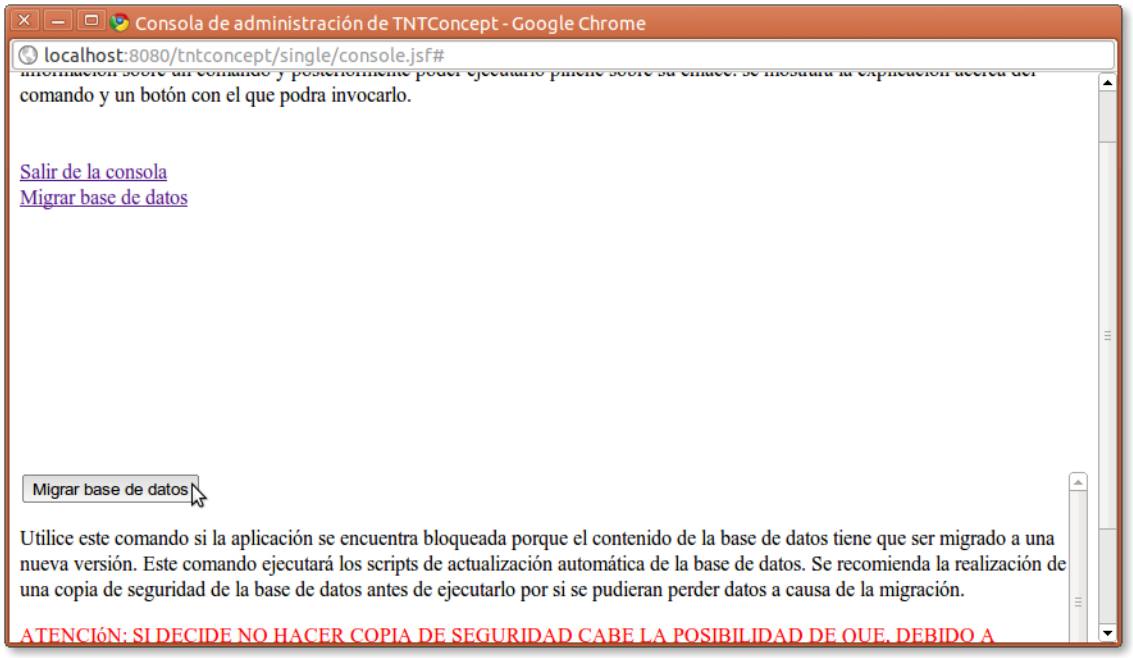

Figura A.2: Ejecutar la migración

Después de un par de minutos, aparecerá el resultado corroborando la migración satisfactoria (figura A.3).

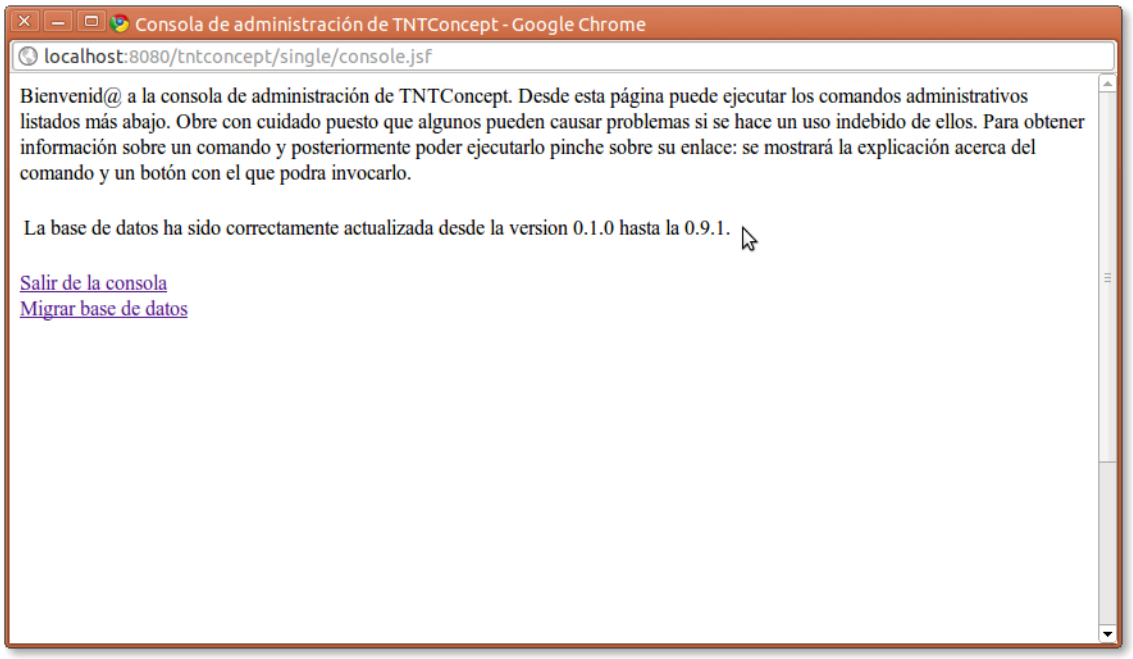

Figura A.3: Migración completada

APÉNDICE B

Manual de usuario

Una vez completada la instalación, pasamos a probarla. Para ello, abrir un navegador y acudir a `http://localhost:8080/tntconcept/`. Se puede observar que ésta se abre en ventana nueva maximizada, para mayor comodidad (figura B.1).

Figura B.1: Pantalla de login y tablón de noticias

El login por defecto es `admin` con la contraseña `adminadmin`. En la portada se puede observar la existencia de un **Tablón de anuncios** público. Sirve para publicar noticias generales de interés

para los empleados o usuarios de la aplicación. Aquí aparecen las noticias cuya categoría se ha definido como *pública*, ya que pueden ser vistas por usuarios que aún no han accedido (iniciado sesión) a la aplicación. Las noticias de otras categorías se pueden consultar en la siguiente pantalla justo después de iniciar la sesión (figura B.2).

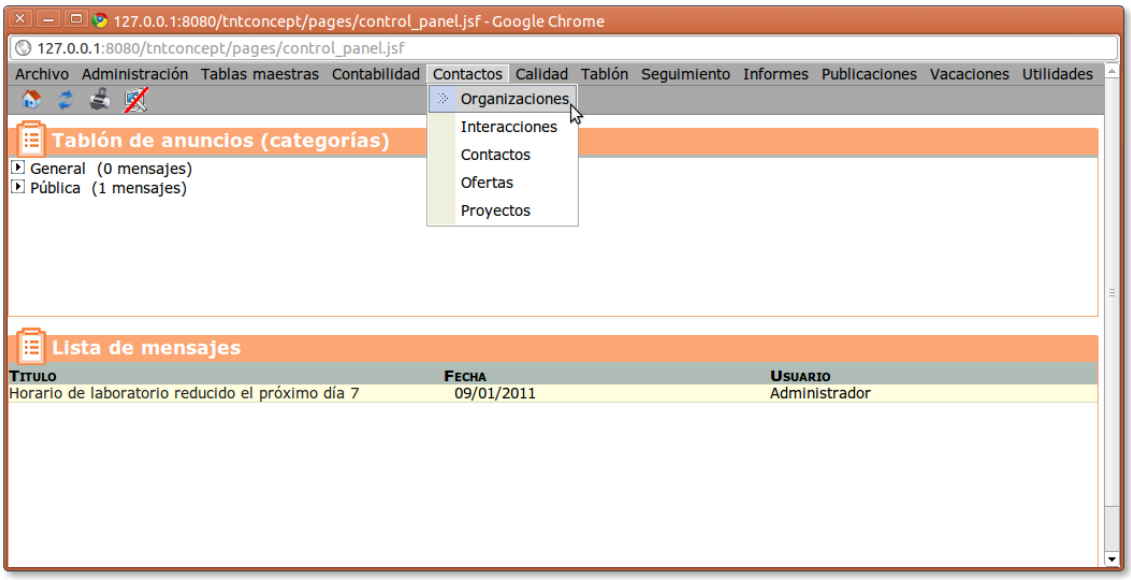

Figura B.2: Tablón de noticias interno

B.1. Gestión de relaciones con el cliente

Bajo este apartado, la aplicación contempla diversos aspectos en las relaciones con el cliente, ofreciendo una cobertura completa del paradigma CRM.

Se encuentra compuesto de los siguientes módulos:

- Gestor de organizaciones
- Gestor de contactos
- Gestor de interacciones
- Gestor de ofertas (presupuestos)

Podemos probar a realizar una iteración completa de todo este flujo de trabajo, para ver cómo los elementos se interrelacionan entre sí.

Ejemplo Supongamos que en una feria de empresas nos pasan un contacto interesado en nuestros productos. El primer paso es dar de alta la Organización (empresa) a la que pertenece. Para ello acudimos al menú *Contactos » Organizaciones*. En esta pantalla vemos el listado actual de organizaciones registradas en nuestra base de datos. Para crear la nueva organización, pulsamos sobre el botón *Nuevo* de la barra de herramientas. En la siguiente pantalla podemos editar todos los campos con la información que tengamos de la empresa:

- nombre,
- CIF,
- dirección,
- teléfonos,
- página web
- tipo de organización (puede ser cliente y/o proveedor)

Observamos que se contempla el atributo *tipo de organización* aunque en el paradigma CRM sólo se aplica a los clientes, podemos reutilizar toda esta lógica de negocio sobre clientes también para los proveedores.

El esquema general para todas las pantalla de «listados» seguirán el mismo patrón:

- la barra de título,
- la barra de herramientas de listado:
  - **filtrar**: muestra un formulario para especificar criterios de búsqueda para filtrar el listado de elementos.
  - **nuevo**: muestra un formulario igual al de edición con los campos editables para crear un nuevo elemento.
  - **ayuda**: abre una ventana con ayuda contextual para esta pantalla.
- clasificador para filtrar por la letra inicial del campo nombre o similar,
- encabezado con las columnas (ordenables ascendentemente o descendetemente) que muestra el listado en forma de tabla
- lista de elementos, uno por fila. Al hacer clic en la fila se muestra el formulario de edición del elemento en cuestión.
- Paginador, en caso que sea necesario.

A continuación podemos editar los contactos conocidos en aquella empresa. Para ello acudimos al menú *Contactos » Contactos*. De nuevo nos encontramos con el listado de contactos, inicialmente vacío. Creamos uno nuevo haciendo clic en el mismo botón de *Nuevo* de la barra de herramientas para ver la pantalla de edición de contacto nuevo.

Este es el patrón general de las pantallas de edición o creación:

- título,
- herramientas del editor:
  - **guardar**: salva o actualiza el contacto presentado
  - **borrar**: borra este contacto, en el caso que ya se encuentre creado. Si el contacto es de nueva creación, este botón no se muestra.

    - **ir atrás**: navegar hasta la pantalla anterior, que es la del listado.
    - **ayuda**: muestra la ayuda contextual.

- formulario con la secuencia de campos para editar, con su etiqueta y el widget apropiado: fecha, texto, sí/no, etc.

Continuando con nuestro ejemplo: supongamos que el cliente nos llama para pedirnos información sobre precios de un ensayo en concreto. Podemos registrar esta interacción con el cliente creando un nuevo registro en la tabla de interacciones a través de la opción de menú *Contacto » Interacciones*, de forma análoga a lo anterior.

Para crear el presupuesto, vamos a la sección de *Contactos » Ofertas*. Una vez completados todos los campos y guardado, podemos imprimirla haciendo clic sobre el botón **imprimir oferta** que genera un pdf con el detalle del presupuesto para poder enviarlo después por email o por fax.

autentia
real business solutions

Nº de oferta:
1

Nuestra empresa
Calle Alguna s/n,
Chiclana 10200
Tlf. 955111222 - CIF: 1222333B

Oferta 1

OFERTA

Datos

Empresa	**Empresa Cliente 1**
Contacto	Juan Espósito
Estado	Abierta
Última modificación	1/31/11 5:54 PM

Fecha de creación	30-Jan-11
Fecha de validez	01-Mar-11

Descripción: Esta es la descripción de la oferta

Observac.:

Gastos

MANO DE OBRA

ROL	HORAS	COSTE/HORA	SUBTOTAL	IVA (%)	IVA (€)	TOTAL

TOTAL MANO DE OBRA (IVA INCLUIDO): $ 0.00

MATERIALES Y OTROS

MATERIALES Y OTROS	COSTE	IVA (%)	IVA (€)	TOTAL
Papel	$ 10.00	16.00 %	$ 1.60	**$ 11.60**
Tinta	$ 5.00	16.00 %	$ 0.80	**$ 5.80**

TOTAL MATERIALES Y OTROS (IVA INCLUIDO): $ 17.40

TOTAL (IVA INCLUIDO): $ 17.40

Para cualquier duda contactar en
o llamando al **955111222**

Monday 31 January 2011 Página 1 de 1

Figura B.5: Ejemplo: Oferta lista para impresión

B.2. Gestión de trabajos

Este módulo simple gestiona los trabajos corrientes o terminados. Reciben el nombre de *Proyectos* (recordemos que Autentia se dedica a desarrollar **proyectos** de software). Se puede crear un proyecto a partir de una Oferta aceptada, copiando sus propiedades usando para ello el botón *Crear proyecto* que aparece en la barra de herramientas en la edición de la oferta; o bien se puede iniciar un proyecto nuevo completando sus datos.

B.3. Gestión de la contabilidad

La sección que gestiona la *Contabilidad* de la empresa se compone de los siguientes elementos:

- **Facturas** emitidas a clientes, y recibidas de proveedores.
- **Cuentas**
- **Asientos**
- **Asientos periódicos**
- **NOF** (Necesidades Operativas de Fondos)
- **Ratios financieros**

Todos estos son tipos de entidades que se pueden listar, buscar, crear, modificar y borrar, con las pantallas típicas de gestión de cada uno de ellos.

En el caso de la emisión de facturas, existe la utilidad de buscar conceptos a incluir en la factura dados la organización cliente y un rango de fechas, de esta forma es fácil asegurar la trazabilidad entre costes de proyectos y costes facturados.

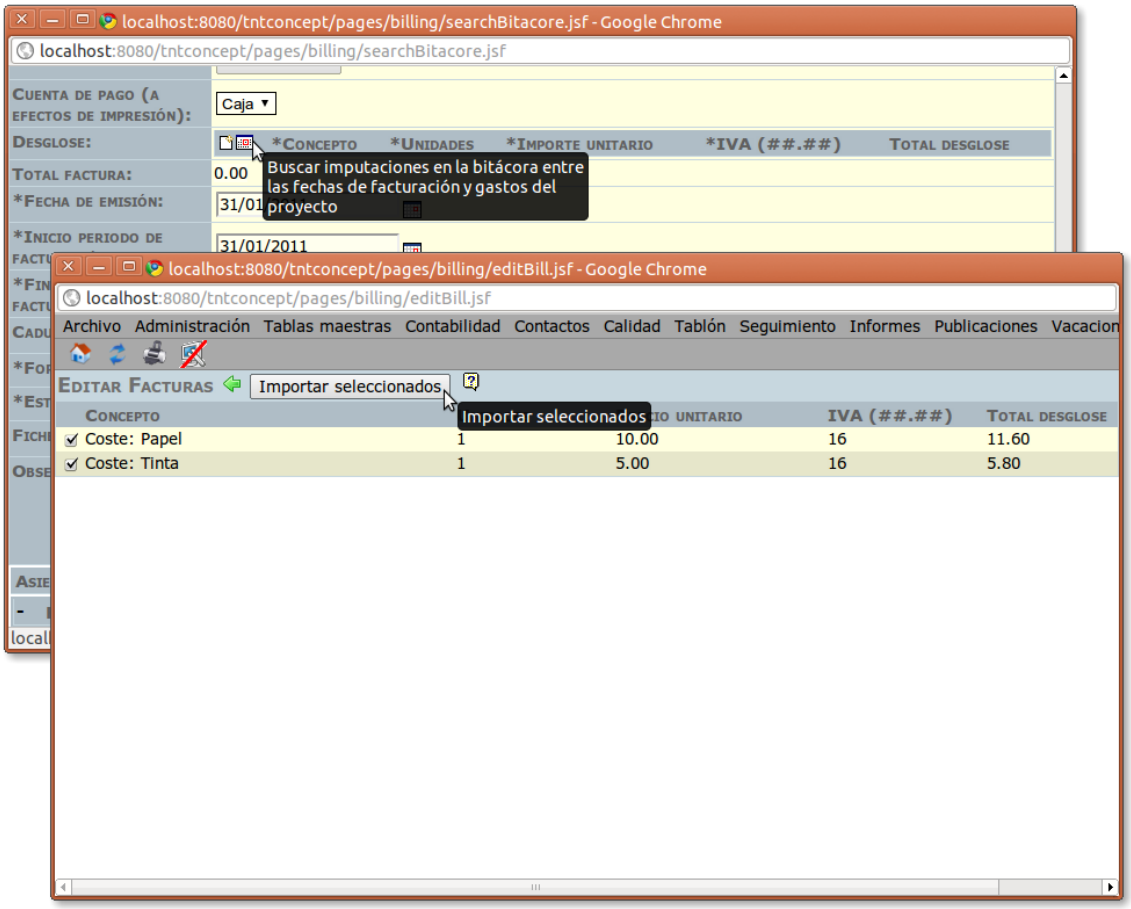

Figura B.6: Buscar costes para imputar a facturas emitidas

B.4. Sistema de información estadística

El software incluye un potente conjunto de informes para obtener información acerca de los clientes, proyectos, facturación, rendimiento de los empleados, etc. Se agrupan bajo el epígrafe denominado *Informes* como opción del menú de la aplicación, y contiene los siguientes grupos:

- Generales
- Actividad
- Facturación
- Proyectos
- Interacciones
- Organizaciones
- Ofertas
- Personales

Los informes *Personales* se refieren a la carpeta de plantillas de informes personales configurada en la propiedad de la aplicación `pathReports`. De esta forma los informes son extensibles en la medida que el diseñador de informes pueda generarlos, como se verá más adelante, con la herramienta **iReport** de la biblioteca en Java de generación de informes **Jasperreports**.

Los informes se pueden obtener en los formatos soportados por la biblioteca, a saber:

- PDF
- CSV
- HTML
- RTF
- XLS
- ODT

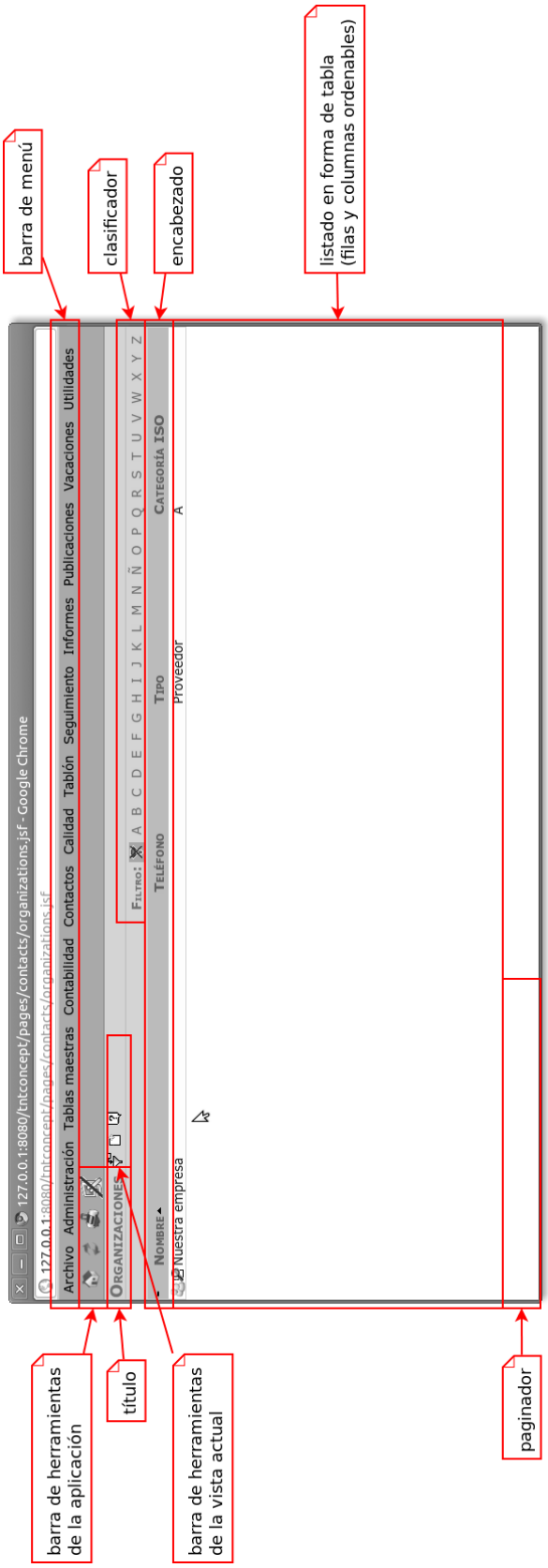

Figura B.3: Pantalla de listado de organizaciones

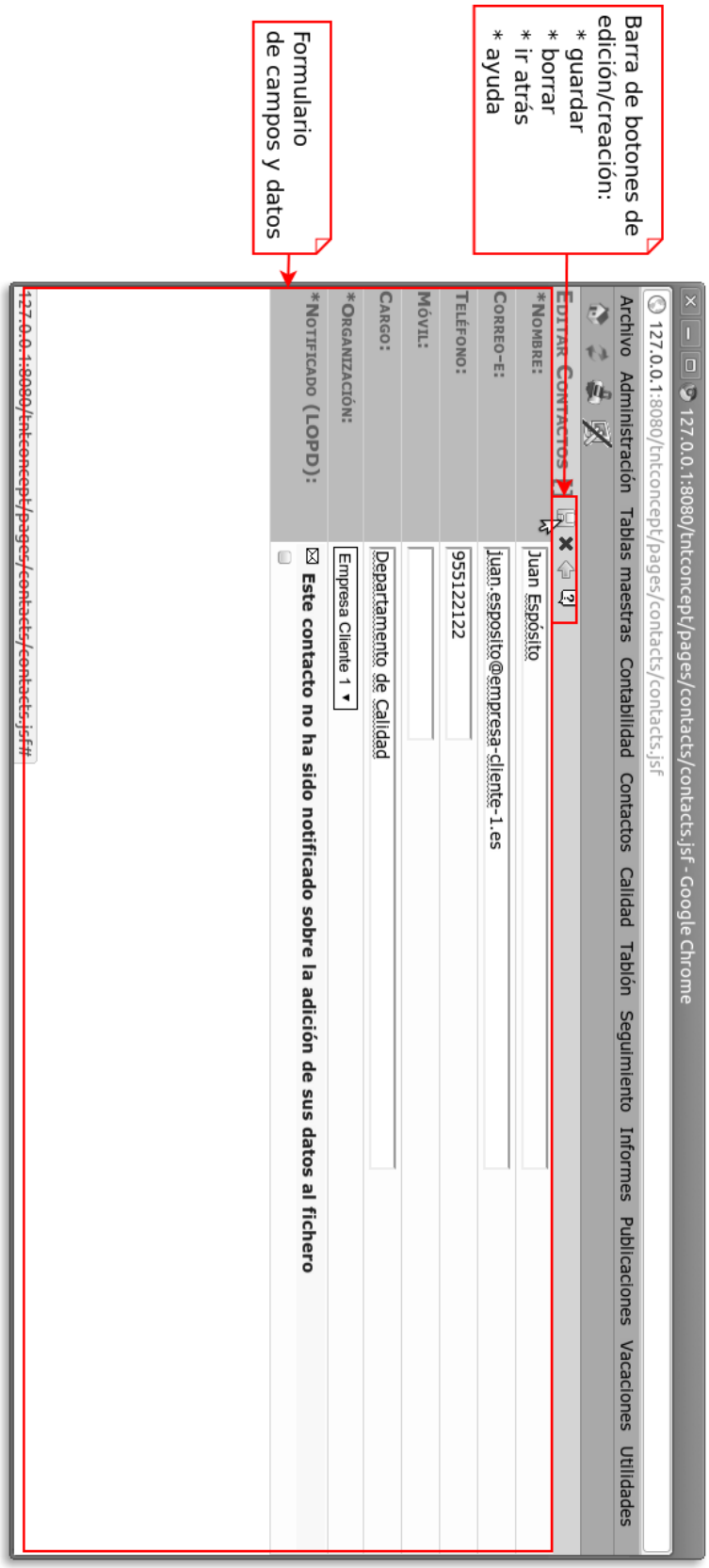

Figura B.4: Pantalla de creación/edición del nuevo contacto

APÉNDICE C

Estudio de soluciones alternativas

Por petición del cliente, el sistema a crear debe construirse en base a alguna de las soluciones de código abierto disponibles, ya que el problema a solucionar no es nuevo, y comenzar una solución desde cero conllevaría un mayor esfuerzo de tiempo y dinero. De manera que el primer paso es identificar software existente similar al deseado, y compararlo.

En particular, hay que valorar tanto el aspecto *funcional* –es decir, la adecuación del software en su estado actual a las necesidades de la empresa–, como el aspecto *técnico* y la viabilidad o las facilidades que se proporcionan para efectuar las modificaciones requeridas: estructura y organización del código fuente (para su mantenibilidad y entendimiento), esquema de la base de datos inicial, tecnología subyacente (bibliotecas y sus dependencias) y la antigüedad de éstas.

Además, al tratarse de aplicaciones de código abierto, es muy importante que exista una interesante comunidad de desarolladores alrededor.

Las aplicaciones que se probaron fueron, todas de código abierto y licencia libre:

- OpenbravoERP [19]
-]project-open[[23]
- TNTConcept [2]
- EBIneutrino [6]
- TinyERP [27]
- ck-erp [4]

Cada una de estas aplicaciones, las descargamos, las instalamos según las instrucciones con su configuración por defecto y las probamos.

Tendremos en cuenta los siguientes criterios para valorar cada una:

- Tamaño de la comunidad de desarrolladores.

  Se puede contar el número de parches subidos al sistema de seguimiento de errores, frecuencia de mensajes en la lista de correo o foro, etc. Es importante para contar con un proyecto base que se encuentre en activo desarrollo para contar con actualizaciones de seguridad o de funcionalidades mejoradas; así como poder encontrar soporte técnico.

- Calidad de la traducción al castellano.

  Sería interesante también poder contar con un sistema base bien traducido, para no tener que traducirlo todo además de realizar las modificaciones necesarias, con sus traducciones.

- Tecnologías base.

  Debe usar lenguajes de programación y tecnologías que no estén obsoletas, y de las que se pueda encontrar documentación y soporte.

- Cliente web.

  La aplicación debe contar con, al menos, una interfaz web desde la que interactuar con el sistema.

- Módulos reutilizables.

  Hay que elegir un proyecto base que también cumpla en mayor parte con los requisitos especificados, o que se puedan modificar.

Pasamos a comentarlas brevemente a continuación.

TinyERP. Tiene dos ejecutables, un servidor propio escrito en Python y un cliente, también Python, de escritorio, usando Gtk+ como biblioteca gráfica. Por ser una aplicación de escritorio, queda descartado, ya que se busca una aplicación web.

ck-erp. Es un conjunto de módulos que se añaden a una instalación de Drupal, en PHP. Sin embargo, la comunidad de desarrolladores es pequeña, y la traducción al español del software es muy deficiente. Se puede ver una demo pública en [4], usando las credenciales `demouser/demouser`.

EBIneutrino. Es software libre la versión comunitaria, pero existe una versión de pago completa. Está programada en lenguaje Java usando Swing para producir el cliente gráfico de escritorio. Al carecer de interfaz web, tampoco encaja en las necesidades. Además, la versión que se probó, sólo contemplaba las funciones de CRM, con lo que habría que añadir bastantes módulos.

]project-open[. Basado en un framework de desarrollo de aplicaciones web llamando OpenACS [18], está mas orientado a desarrollo de proyectos que a la venta de productos. Está programado en Tcl, un lenguaje completamente desconocido para mí, y que tampoco le interesó al cliente. La comunidad de desarrolladores es pequeña, lo que dificultaba también encontrar soporte en el caso que fuera necesario. Y no existe traducción ninguna al castellano.

Las dos aplicaciones que quedan, son las que estudiamos más profundamente y describimos en las siguientes secciones. Ambos satisfacen completamente los criterios mencionados anteriormente; si bien OpenbravoERP tiene demasiados módulos que no se aplican al negocio de la empresa y tendríamos que dedicar más tiempo a ver cómo quitar los módulos que sobran que a añadir los que faltan.

C.1. OpenbravoERP

OpenbravoERP [19] disfruta de una licencia Mozilla Public License (MPL), aceptado por la Open Source Initiative como licencia de código abierto. Se compone de distintas partes programadas en lenguaje Java; y necesita para su correcto funcionamiento, como mínimo:

- PostgreSQL8.1 u Oracle 10.2
- Apache Tomcat 5.5
- SUN JDK 5

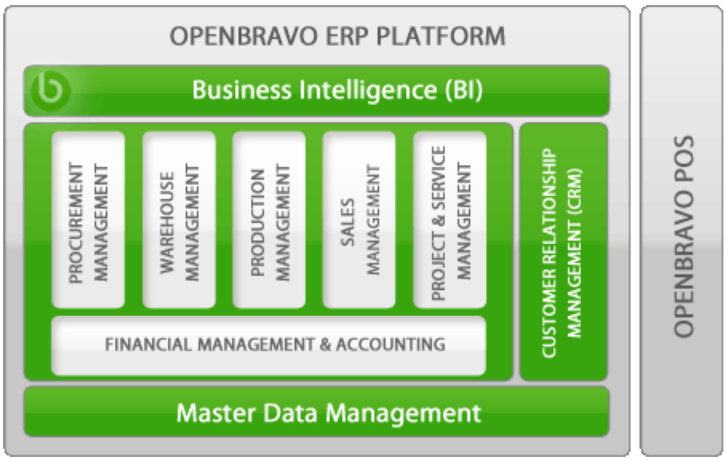

Figura C.1: Componentes de OpenbravoERP

Su instalación no es inmediata pero es sencilla. Estudiamos en su momento la versión v2.35mp1. Se puede encontrar el repositorio Mercurial en la url [20]

C.1.1. Instalación desde el código fuente

Existe un guión de instalación, pero si tenemos intención de hacer modificaciones sobre el software, la documentación nos recomienda hacerla desde el código fuente, en el entorno de desarrollador.

C.1.1.1. Requisitos del sistema

Primero tenemos que instalar las dependencias de la aplicación, que son:

- El SDK y JVM de la máquina Java de Sun Microsystems, Inc. Como mínimo, se tiene que instalar JDK5 aunque aquí lo hemos probado con JDK6.
- La base de datos Oracle 10g. Utilizamos aquí la versión xe (Express Edition) que es gratis para usos no comerciales, y del cual existen paquetes *rpm* y *deb* para Fedora, Ubuntu, Debian, Red Hat, Suse, etc., para una instalación más sencilla.

No detallamos estos pasos ya que dependen de la distribución que estemos usando.

C.1.1.2. Configuración y compilación

Una vez descomprimido el `.tar.gz` de Openbravo 2.35mp1, nos situamos en una terminal dentro y ejecutamos el setup:

```
$ ant -f build.xml.template setup
```

En la primera pantalla, se nos solicita las rutas de instalación. Para hacerlo más sencillo, podemos cambiar la subcadena `/opt/AppsOpenbravo` por la ruta a la carpeta donde tenemos el código fuente.

Seguidamente, introducimos los datos de conexión a la base de datos, con lo que finaliza el asistente de configuración. Esto nos ha creado un fichero `build.xml` con las rutas configuradas para los objetivos de Ant.

El siguiente paso es crear y completar la base de datos, compilar todo el código fuente y empaquetar la aplicación. Todo esto se consigue ejecutando el siguiente comando:

```
$ ant install.source
```

El resultado es el paquete de la aplicación J2EE localizado en `lib/openbravo.war`.

Podemos incluir datos de muestra usando el siguiente objetivo Ant:

```
$ ant import.database.sampledata
```

C.1.2. Estructura del código y arquitectura

El ecosistema de Openbravo se compone de distintos módulos:

- core: Paquetes básicos con interfaces para la base de datos, Sqlc, XmlEngine, y otras utilidades.
- db: configuración del *pool* de la base de datos.
- trl: manejo de traducciones de internacionalización.
- wad: Wizard for Application Dictionary, un asistente para construir ventanas y solapas siguiendo la definición del diccionario de la aplicación.

Openbravo sigue la metodología MDD (Model-driven Development), un enfoque de programación orientado al modelo. Y los modelos son los que se definen en la base de datos, como una especie de *metadatos* de la aplicación. En Openbravo, la configuración de ventanas, pantallas, formularios, listados, procesos, botones y acciones, y todo este tipo de recursos se pueden definir en el apartado de Diccionario de la Aplicación (AD), junto con piezas de programación definidas en el árbol del proyecto.

A través de un complejo sistema de generación de código, el sistema se compila a sí mismo de forma que se puede ir completando a medida que el negocio crece. Por eso se dice que es de rápida implantación, pero lenta y progresiva adaptación.

No obstante, el código generado sigue el patrón MVC separando la visualización, de los datos y de su tratamiento, en capas que colaboran entre sí pero desacopladas.

Tiene demasiados pasos y códigos intermedios que, por un lado reducen la tasa de errores humanos al existir gran cantidad de código auto-generado, pero como contrapartida, es demasiado amplio para este caso concreto, y por eso lo desechamos. Aunque es bastante interesante como objeto de estudio.

C.1.3. Uso

La interfaz de usuario es simple: tiene un menú de navegación en forma de árbol en un lateral, y en el marco principial que cambia entre un formulario de entrada de datos para un registro, o un listado de registros. Se pueden editar registros relacionados de uno a muchos navegando por las pestañas situadas debajo de la barra de herramientas. La barra de herramientas tiene las funciones básicas sobre registros como crear nuevo, borrar, guardar cambios, y navegar al registro anterior/siguiente y primero/último.

En la figura C.2 podemos ver como ejemplo la pantalla de edición de los datos de Cliente de un Tercero. Se observa la cantidad de campos que existen, con una alta normalización.

Al tratarse de un ERP genérico, trata muchos aspectos, algunos pueden no aplicar para el caso concreto:

- Gestión de terceros: clientes y proveedores (CRM).
- Gestión de productos: compras, inventario, almacenes, aprovisionamiento, rutas de transporte, informes.
- Gestión de compras: pedidos, albaranes, facturas, informes.
- Gestión de ventas: pedidos, albaranes, facturas, informes, procesos, comisiones, marketing.
- Gestión de cobros y pagos: remesas, liquidación, movimientos bancarios.
- Gestión contable: asientos, cuentas, presupuestos, informes.
- Gestión de proyectos y servicios, obras.

C.2. Autentia TNTConcept

TNTConcept es un producto tanto gratuito como abierto, distribuido bajo la licencia GPLv2. La propia empresa creadora del software, Autentia, la define como una «herramienta para la Gestión de la Eficacia Operativa (GEO)» [2]. Incluye todos los aspectos cubiertos por el modelo Customer Relationship Mangement (CRM), la gestión de contabilidad y finanzas, y parcialmente la visión de Enterprise Resource Planing (ERP).

La versión estudiada fue TNTConcept 0.9.1. Los requisitos que ésta necesita para funcionar son bastante alcanzables:

- MySQL 5
- Apache Tomcat 5.5
- SUN JDK 5

Este producto fue finalmente el elegido como base para la realización de este Proyecto. El manual de instalación se encuentra en el apéndice A y el de usuario en el apéndice B.

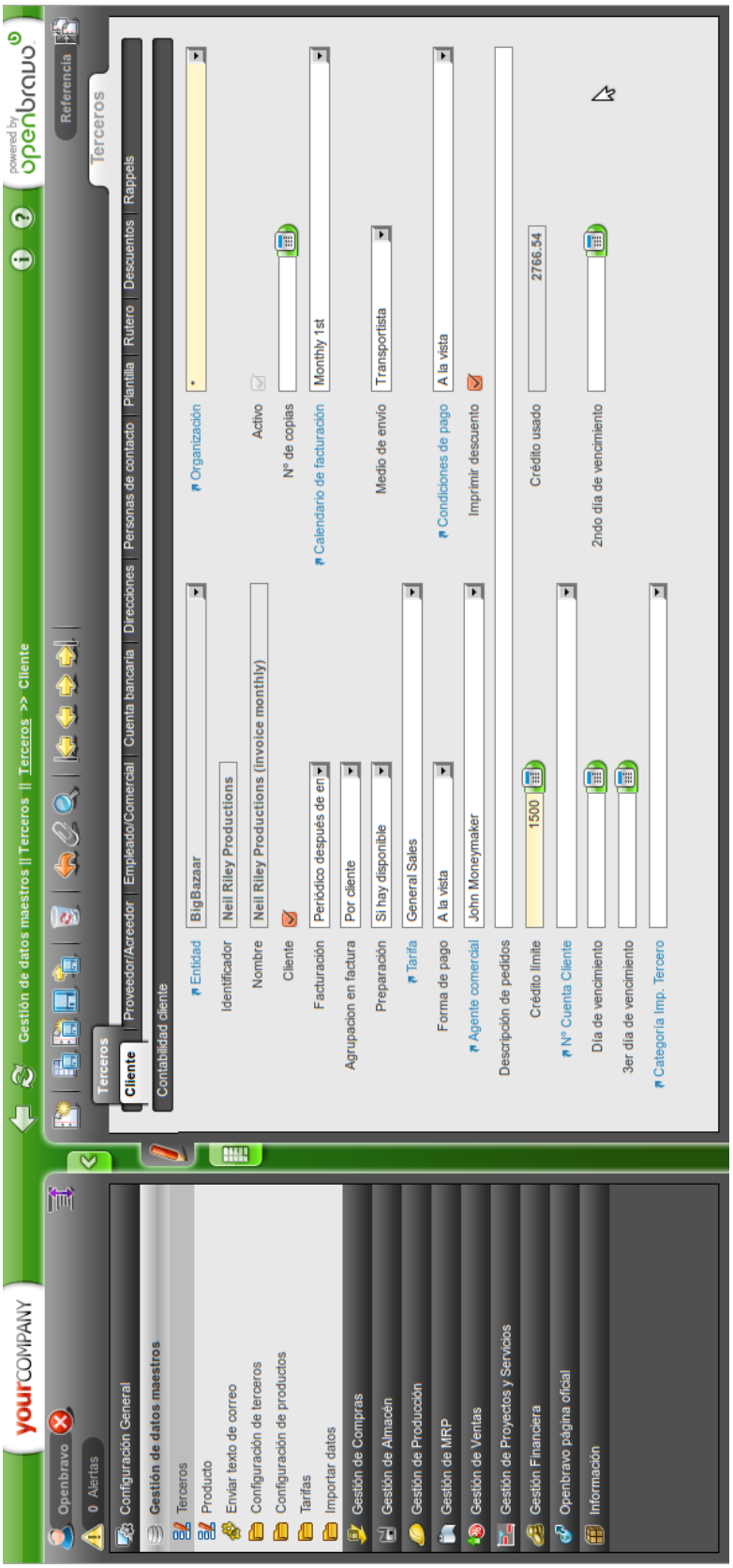

Figura C.2: Openbravo: Edición de Tercero Cliente

Referencias bibliográficas

[1] Gerardo Aburruzaga García and Francisco Palomo Lozano. Make. un programa para controlar la recompilación. `http://www.uca.es/softwarelibre/publicaciones/make.pdf` [Última visita: febrero de 2008].

[2] Autentia, S.L. Web de TNTConcept. `http://tntconcept.sourceforge.net/` [Última visita: febrero de 2012].

[3] B. W. Boehm. *Verifying and Validating Software Requirements and Design Specifications*, volume 1. IEEE, 1984.

[4] ck-erp. `http://sourceforge.net/projects/ck-erp/` [Última visita: febrero de 2008].

[5] William Crawford and Jonathan Kaplan. *J2EE Design Patterns*. O'Reilly Media, 2003.

[6] EBIneutrino. `http://sourceforge.net/projects/ebineutrino/` [Última visita: febrero de 2008].

[7] David Flanagan. *Java in a Nutshell, 5th ed.* O'Reilly, 2005.

[8] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[9] Marty Hall. JSF 2.0 Tutorials. `http://www.coreservlets.com/JSF-Tutorial/jsf2` [Última visita: mayo de 2008].

[10] David R. Heffelfinger. *JasperReports for Java Developers*. Packt Publishing, 2006.

[11] IEEE. *IEEE Recommended Practice for Software Requirements Specification (IEEE Std 830-1998)*. IEEE Computer Society, 1998.

[12] iReport. `http://jasperforge.org/projects/ireport` [Última visita: octubre de 2011].

[13] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Prentice Hall, 2004.

[14] Luke Grzegorz Maciak. LaTeX: Squeezing the Vertical White Space. http://www.terminally-incoherent.com/blog/2007/09/19/latex-squeezing-the-vertical-white-space/ [Última visita: diciembre de 2011].

[15] MÉTRICA versión 3. http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es [Última visita: febrero de 2012].

[16] NetBeans. http://www.netbeans.org/ [Última visita: octubre de 2011].

[17] Tim O'Brien. *Maven: The Definitive Guide*. O'Reilly Media, 2008.

[18] OpenACS. http://www.openacs.org/ [Última visita: febrero de 2008].

[19] OpenbravoERP. http://www.openbravo.com/ [Última visita: febrero de 2012].

[20] OpenbravoERP. Repositorio mercurial. https://code.openbravo.com/erp/stable/2.3x/ [Última visita: diciembre de 2011].

[21] C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version Control with Subversion*. O'Reilly Media, 2008.

[22] Planner. http://live.gnome.org/Planner [Última visita: febrero de 2012].

[23]]project-open[. http://sourceforge.net/projects/project-open/ [Última visita: febrero de 2008].

[24] Chris Schalk, Ed Burns, and James Holmes. *JavaServer Faces: The Complete Reference*. McGraw-Hill Osborne Media, 2006.

[25] Selenium IDE. http://seleniumhq.org/ [Última visita: febrero de 2012].

[26] StatSVN. http://www.statsvn.org/ [Última visita: octubre de 2011].

[27] TinyERP. http://sourceforge.net/projects/tinyerp/ [Última visita: enero de 2012].

[28] TITANIA, Ensayos y Proyectos Industriales, S.L.U. http://www.titania.aero/ [Última visita: diciembre de 2011].

[29] Trac. http://trac.edgewall.org/ [Última visita: febrero de 2010].

[30] Zubin Wadia, Martin Marinschek, Hazem Saleh, and Dennis Byrne. *The Definitive Guide to Apache MyFaces and Facelets*. Apress, 2008.

[31] Wikibooks Community. LaTeX Wikibook. http://en.wikibooks.org/wiki/latex [Última visita: enero de 2012].

[32] Wikipedia Community. Wikipedia en Español. http://es.wikipedia.org/wiki [Última visita: febrero de 2012].

[33] Wikipedia Community. Wikipedia en Inglés. http://en.wikipedia.org/wiki [Última visita: febrero de 2012].

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide

which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.